HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Jens Reißner

Moderne Gradientenverfahren für den Einsatz in technischen Aufgabenstellungen und deren Anwendung auf ein- und multikriterielle Optimierungsaufgaben

2013

Fakultät **Mathematik Naturwissenschaften**
Informatik

BACHELORARBEIT

Moderne Gradientenverfahren für den Einsatz in technischen Aufgabenstellungen und deren Anwendung auf ein- und multikriterielle Optimierungsaufgaben

Autor:
Jens Reißner

Studiengang:
Angewandte Mathematik

Seminargruppe:
MA10w1-B

Erstprüfer:
Prof. Dr. Ullrich Griesbach

Zweitprüfer:
M. Sc. Steffen Kux

Mittweida, September 2013

Bibliografische Angaben

Reißner, Jens: Moderne Gradientenverfahren für den Einsatz in technischen Aufgabenstellungen und deren Anwendung auf ein- und multikriterielle Optimierungsaufgaben, 57 Seiten, 32 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik Naturwissenschaften Informatik

Bachelorarbeit, 2013

Satz: LaTeX

Referat

In dieser Arbeit werden zwei gradientenbasierte Verfahren zur Lösung nichtlinearer Optimierungsaufgaben vorgestellt. Neben der Theorie der Verfahren werden konkrete algorithmische Umsetzungen beschrieben und Hinweise zur Implementierung gegeben. Ein Vergleich der Algorithmen mit gängigen Optimierungsverfahren erfolgt anhand einiger ausgewählter Testfunktionen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Vorwort

Diese Bachelorarbeit zum Thema "*Moderne Gradientenverfahren für den Einsatz in technischen Aufgabenstellungen und deren Anwendung auf ein- und multikriterielle Optimierungsaufgaben*" entstand in Rahmen meines Bachelorstudiums an der Hochschule Mittweida (FH) in Zusammenarbeit mit der IAV GmbH, Chemnitz.

Optimierung spielt bei der Entwicklung technischer Bauteile oder Prozesse eine immer wichtigere Rolle. Um die begrenzten Ressourcen bestmöglich auszunutzen und Produktionsabläufe zu beschleunigen, nutzt man verstärkt die Methoden der mathematischen Optimierung. Praktische Problemstellungen werden hierzu mathematisch abstrahiert und mit einem geeigneten Optimierungsverfahren gelöst.
Da bei technischen Anwendungen häufig stetig differenzierbare Ziele, wie z.B. Summe der Fehlerquadrate, optimiert werden müssen, eignen sich gradientenbasierte Optimierungsverfahren zum schnellen und effizienten Auffinden einer Optimallösung. Die Arbeit mit Gradienteninformationen erlaubt dabei eine gezielte Suche nach Minima und führt zu einer hohen Genauigkeit der erhaltenen Lösungen.
Auch für den komplexen Fall der Mehrzieloptimierung bieten gradientenbasierte Methoden eine interessante Alternative zu den dort etablierten evolutionären Algorithmen. Hierbei steht weniger die Approximation der Lösungsmenge durch hunderte Punkte, sondern viel mehr das Interesse an einzelnen Lösungspunkten mit hoher Genauigkeit im Vordergrund.

In dieser Arbeit werden zwei iterative Methoden zur Lösung der oben beschriebenen Optimierungsprobleme vorgestellt, deren Iterationsverlauf durch Gradienteninformationen bestimmt wird. Neben dem klassischen BFGS-Verfahren zur Optimierung einkriterieller Probleme wird mit dem MGDA-Verfahren ein moderner Ansatz der multikriteriellen gradientenbasierten Optimierung beschrieben.

1 Grundlagen der nichtlinearen Optimierung

In diesem Kapitel soll ein Einblick in die Theorie der ein- und multikriteriellen Optimierung nichtlinearer Funktionen gegeben werden. Neben Definitionen und wichtigen Grundbegriffen werden einige Elemente der linearen Algebra eingeführt, die für die im weiteren Verlauf der Arbeit beschriebenen Optimierungsverfahren von Bedeutung sind. Weiter wird beim Begriff der Optimalität auf Gemeinsamkeiten und Unterschiede zwischen ein- und multikriteriellen Aufgabenstellungen eingegangen und es werden kurz zwei Ansätze zur Umformung eines multikriteriellen in ein einkriterielles Problem beschrieben.

Im Folgenden werden bei theoretischen Aussagen und Beispielen stets nur Minimierungsprobleme betrachtet. Dies ist ohne Beschränkung der Allgemeinheit möglich, da ein Maximierungsproblem $f(\mathbf{x}) \to max$ äquivalent zum Minimierungsproblem $-f(\mathbf{x}) \to min$ ist. Da die in dieser Arbeit beschriebenen Verfahren mit Ableitungen der Zielfunktionen arbeiten, sei weiter vorausgesetzt, dass sowohl die Zielfunktionen als auch die Nebenbedingungen zweimal stetig differenzierbar sind.

1.1 Die einkriterielle Optimierungsaufgabe

Der folgende Abschnitt beschäftigt sich mit den theoretischen Grundlagen der einkriteriellen Optimierung. Eine allgemeine Definition des Problems wird in [GK99] gegeben.

Definition 1.1 Unter einer einkriteriellen Optimierungsaufgabe versteht man folgendes Problem: Minimiere $f : D \to \mathbb{R}$ unter den Nebenbedingungen $\mathbf{x} \in D$. Dabei wird D als *zulässiger Bereich* bezeichnet.

Definition 1.2 Der *zulässige Bereich* $D \subseteq \mathbb{R}^n$ ist die Menge aller Punkte, welche die Nebenbedingungen erfüllen.

Falls $D = \mathbb{R}^n$ spricht man von einer unrestringierten Optimierungsaufgabe. Andernfalls, wenn $D \neq \mathbb{R}^n$, handelt es sich um eine restringierte Optimierungsaufgabe.

Im Falle der restringierten Optimierung wird der zulässige Bereich D durch Gleichheitsnebenbedingungen $g_l(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, \quad l = 1, ..., L$ sowie Ungleichheitsnebenbedingungen $h_m(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, \quad m = 1, ..., M$ beschrieben. Die einkriterielle Optimierungsaufgabe

kann damit auf folgende Weise notiert werden:

$$\begin{aligned} f(\mathbf{x}) &\to min \\ g_l(\mathbf{x}) &= 0, \quad l = 1, ..., L \\ h_m(\mathbf{x}) &\leq 0, \quad m = 1, ..., M \\ \mathbf{x} &\in \mathbb{R}^n. \end{aligned} \tag{1.1}$$

Auch an dieser Stelle ist es ohne Beschränkung der Allgemeinheit möglich, Ungleichungen der Form $h_m(\mathbf{x}) \geq 0$ äquivalent in $-h_m(\mathbf{x}) \leq 0$ umzuformen.

Die Optimierungsaufgabe (1.1) wird als *nichtlinear* bezeichnet, wenn mindestens eine der Funktionen $f(\mathbf{x})$, $g_l(\mathbf{x})$, $h_m(\mathbf{x})$ nichtlinear ist.

Definition 1.3 Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine Funktion. f heißt *linear*, falls

$$f(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{y}), \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \quad \forall\, \alpha, \beta \in \mathbb{R}.$$

Andernfalls wird sie als *nichtlinear* bezeichnet.

Bemerkung 1.4 Bei vielen Optimierungsproblemen werden für die Parameter $\mathbf{x} = (x_1, ..., x_n)^T$ Intervallnebenbedingungen ("box constraints") der Form $x_u < x < x_o, \quad x_u, x_o \in \mathbb{R}$ vorgegeben. Diese können in zwei Ungleichheitsnebenbedingungen $x_u - x < 0$ und $x - x_o < 0$ umgewandelt werden. Im Weiteren werden sie auf diese Art in die allgemeine Aufgabenstellung integriert.

Die Lösungen des Optimierungsproblems (1.1) lassen sich wie folgt klassifizieren (vgl.[JS00])

Definition 1.5 Sei $f : D \to \mathbb{R}$ eine beliebige Funktion mit $D \subseteq \mathbb{R}^n$. Ein Punkt $\mathbf{x}^* \in D$ heißt *lokales Minimum* von f, falls es eine Umgebung $U \subseteq D$ von $\mathbf{x}^*$ gibt, sodass $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ für alle $\mathbf{x} \in U$ gilt. Ein Punkt $\mathbf{x}^* \in D$ heißt *globales Minimum* von f, falls $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ für alle $\mathbf{x} \in D$ gilt.

Beispiel 1.6 Minima einer eindimensionalen Funktion

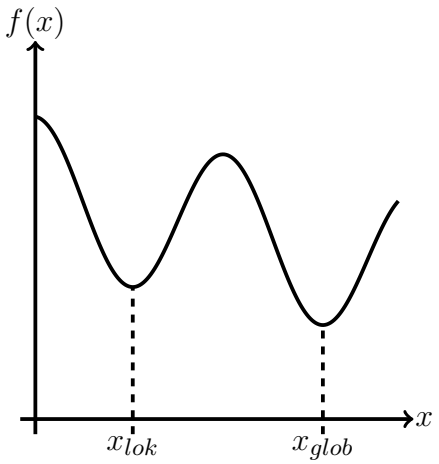


Abbildung 1.1: Eindimensionale Funktion mit lokalem und globalem Minimum

In Abbildung 1.1 stellt x_{lok} ein lokales Minimum dar, da es in der lokalen Umgebung um x_{lok} keine Punkte mit $f(x) < f(x_{lok})$ gibt. Der Punkt x_{glob} hingegen ist lokales und gleichzeitig globales Minimum, da für alle anderen Punkte des Definitionsbereiches $f(x) > f(x_{glob})$ gilt.

Um Bedingungen für die Optimalität eines Punktes $\mathbf{x} \in D$ anzugeben, benötigt man zunächst noch einige Begriffe aus der Differentialrechnung.

Definition 1.7 Der *Gradient* $\nabla f(\mathbf{x})$ einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ ist der Vektor der ersten partiellen Ableitungen.

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right)^T \tag{1.2}$$

Definition 1.8 Die *Jacobi-Matrix* $J(\mathbf{x})$ einer Vektorfunktion $f : \mathbb{R}^n \to \mathbb{R}^m$ ist die Matrix der ersten partiellen Ableitungen.

$$J(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{\partial f_m}{\partial x_1} & \dots & \dots & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \tag{1.3}$$

Definition 1.9 Die *Hesse-Matrix* $\nabla^2 f(\mathbf{x})$ einer skalaren Funktion $f : \mathbb{R}^n \to \mathbb{R}$ ist die Matrix der zweiten partiellen Ableitungen.

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{1.4}$$

Mit Hilfe der Definitionen 1.5 - 1.9 ist es nun möglich, Bedingungen für die Optimalität eines Punktes $\mathbf{x} \in D$ zu formulieren.

Satz 1.10 *(notwendige Bedingung) Sei $\boldsymbol{x}^*$ lokales Minimum einer zweimal stetig partiell differenzierbaren Funktion $f : \mathbb{R}^n \to \mathbb{R}$. Dann gilt*

$$\nabla f(\boldsymbol{x}^*) = 0. \tag{1.5}$$

Punkte $\mathbf{x}^*$ mit $\nabla f(\mathbf{x}^*) = 0$ werden als *stationäre Punkte* bezeichnet.

Beweis: (vgl. [Wei10]) Für beliebiges $\mathbf{v} \in \mathbb{R}^n$ mit $\mathbf{v} \neq 0$ und $t \in \mathbb{R}$ sei

$$\varphi(t) := f(\mathbf{x}^* + t\mathbf{v})$$

in einer Umgebung von $t = 0$ stetig differenzierbar. Ferner habe φ in $t = 0$ ein Extremum. Mit der Kettenregel gilt dann

$$\begin{aligned} \varphi'(t) &= \nabla f(\mathbf{x}^* + t\mathbf{v})^T \mathbf{v} \\ 0 = \varphi'(0) &= \nabla f(\mathbf{x}^*)^T \mathbf{v} \end{aligned}$$

Da $\mathbf{v} \neq 0$ beliebig gewählt wurde, ist $\nabla f(\mathbf{x}^*)^T = 0$. □

Satz 1.11 *(hinreichende Bedingung) Sei $\nabla f(\boldsymbol{x}^*) = 0$ und $\nabla^2 f(\boldsymbol{x}^*)$ positiv definit. Dann ist $\boldsymbol{x}^*$ lokales Minimum von f.*

Beweis: (vgl.[Wei10]) Sei $\nabla^2 f(\mathbf{x}^*)$ positiv definit. Da f zweimal stetig differenzierbar ist, ist $\nabla^2 f(\mathbf{x})$ auch in einer Umgebung $U_\epsilon(\mathbf{x}^*)$ für hinreichend kleine $\epsilon > 0$ positiv definit. Für die Taylor-Entwicklung von $\mathbf{x} \in U_\epsilon(\mathbf{x}^*) \backslash \mathbf{x}^*$ gilt damit

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \nabla f(\mathbf{x}^*)^T (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \nabla^2 f[\mathbf{x}^* + \theta(\mathbf{x} - \mathbf{x}^*)](\mathbf{x} - \mathbf{x}^*), \quad \theta \in (0,1)$$

Mit Bedingung (1.5) gilt der Umgebung von $\mathbf{x}^*$

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T \nabla^2 f[\mathbf{x}^* + \theta(\mathbf{x}-\mathbf{x}^*)](\mathbf{x}-\mathbf{x}^*) > 0, \quad \theta \in [0,1]$$

Da $\mathbf{x}$ beliebig gewählt wurde, folgt aus $f(\mathbf{x}) - f(\mathbf{x}^*) > 0$, dass $\mathbf{x}^*$ lokales Minimum von $f(\mathbf{x})$ ist. □

Neben den numerischen Lösungsmethoden existiert mit der *Lagrange-Methode* ein analytischer Ansatz zur Lösung nichtlinearer Optimierungsprobleme, der für die weiteren Betrachtungen von Bedeutung ist. Hierbei wird das restringierte Optimierungsproblem in ein unrestringiertes Problem umgewandelt.

Definition 1.12 Die Lagrange-Funktion der Optimierungsaufgabe (1.1) ist die Funktion (vgl.[Sch00])

$$L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \sum_{l=1}^{L} \lambda_l g_l - \sum_{m=1}^{M} \mu_m h_m$$

Für ein Minimum dieser Funktion gelten erweiterte nowendige Optimalitätsbedingungen.

$$\begin{aligned}
\nabla_{\mathbf{x}_i} L(\mathbf{x}^*, \lambda^*, \mu^*) &= 0 \quad i = 1, ..., n \\
\nabla_{\lambda_l} L(\mathbf{x}^*, \lambda^*, \mu^*) &= 0 \quad l = 1, ..., L \\
\nabla_{\mu_m} L(\mathbf{x}^*, \lambda^*, \mu^*) &\leq 0 \quad m = 1, ..., M \\
h_m(\mathbf{x}^*)\mu_m^* &= 0 \quad m = 1, ..., M \\
\mu_m^* &\geq 0 \quad m = 1, ..., M \\
\nabla g_l,\ \nabla h_m,\ linear\ unabhängig &\quad \forall l, m : h_m\ aktiv
\end{aligned}$$

Punkte, die diese Bedingungen erfüllen, werden auch als *Kuhn-Tucker-Punkte* bezeichnet. Für genauere Informationen zur Kuhn-Tucker-Theorie sei auf [Obe12] verwiesen.

1.2 Multikriterielle nichtlineare Optimierung

Das Konzept der multikriteriellen Optimierung stellt eine Erweiterung der einkriteriellen Optimierung dar. In der Literatur werden derartige Aufgabenstellungen oft auch als *Vektor- oder Mehrzieloptimierung* bezeichnet. Hierbei wird ein Optimierungsproblem

bezüglich mehrerer Zielfunktionen betrachtet.

$$\begin{aligned} f_k(\mathbf{x}) &\to min, \quad k = 1, ..., K \\ g_i(\mathbf{x}) &= 0, \quad l = 1, ..., L \\ h_j(\mathbf{x}) &\leq 0, \quad m = 1, ..., M \\ \mathbf{x} &\in \mathbb{R}^n \end{aligned} \tag{1.6}$$

Ziel ist es nun, das Problem bezüglich aller Zielfunktionen zu optimieren. Ein einführendes Beispiel soll diesen Sachverhalt verdeutlichen.

Beispiel 1.13 Bei der Anschaffung eines neuen Autos stellt man sich stets die Frage nach dem richtigen Zeitpunkt. Je älter das gebrauchte Fahrzeug, desto geringer ist sein Wiederverkaufswert. Dem entgegen steht die Frage, ob der Kauf eines neuen Fahrzeugs schon notwendig ist, wenn sich das alte noch in einem guten Zustand befindet. Dieser Sachverhalt soll durch die folgende Grafik ausgedrückt werden.

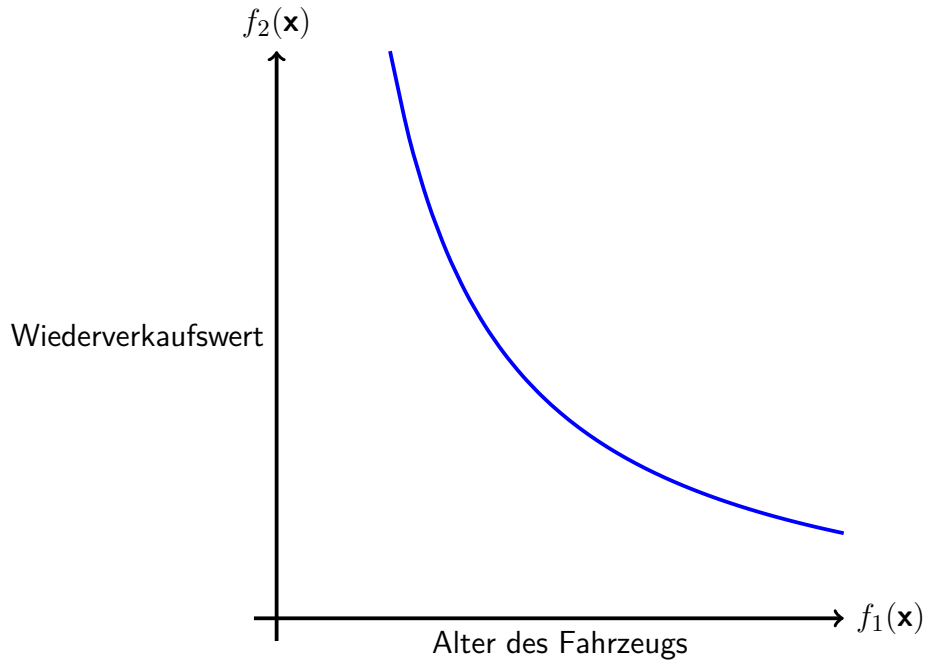


Abbildung 1.2: Verhältnis von Alter und Wiederverkaufswert eines Fahrzeugs

Abbildung 1.2 zeigt ein typisches Optimierungsproblem mit zwei widersprüchlichen Zielen. Wartet man sehr lange mit der Anschaffung eines neuen Fahrzeugs, bekommt man für das Alte nur noch einen Bruchteil des ursprünglichen Kaufpreises. Wer sein Auto häufiger wechselt, kann mit einem höheren Wiederverkaufswert rechnen, muss jedoch auch häufiger den Restbetrag für die Neuanschaffung aufbringen. Man erkennt schnell, dass bei diesem Problem nicht "die" optimale Lösung existiert. Vielmehr zeigt der Graph eine Reihe von Kompromisslösungen, welche im klassischen Sinne nicht miteinander

vergleichbar sind. Verbesserungen bezüglich des einen Ziels führen hier jeweils zu einer Verschlechterung des anderen.

Das Ergebnis einer multikriteriellen Optimierungsaufgabe ist im Allgemeinen kein einzelner Punkt, sondern eine Menge von Kompromisslösungen. Zur Klassifizierung dieser Kompromisse eignet sich die *Dominanzrelation*.

Definition 1.14 Ein Punkt $\mathbf{x} \in \mathbb{R}^n$ *dominiert* einen Punkt $\mathbf{y} \in \mathbb{R}^n$, wenn er bezüglich aller Zielfunktionen gleich gut, sowie bezüglich mindestens einer Zielfunktion besser ist als $\mathbf{y}$.

$$\mathbf{x} \prec \mathbf{y} \Leftrightarrow f_k(\mathbf{x}) \leq f_k(\mathbf{y}) \quad k = 1, ..., K \quad \wedge \quad \exists a \in \{1, 2, ..., K\} \; mit \; f_a(\mathbf{x}) < f_a(\mathbf{y})$$

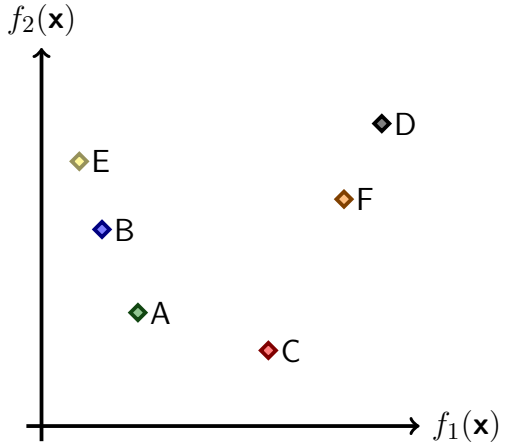


Abbildung 1.3: Auswahl von Punkten innerhalb des Funktionswertebereich

Abbildung 1.3 zeigt eine Auswahl von Punkten innerhalb des Funktionswertebereiches einer zweikriteriellen Minimierungsaufgabe. Punkt D wird von allen anderen Punkten dominiert, da er bezüglich beider Zielfunktionen schlechtere Werte aufweist, als alle anderen Punkte. Punkt F dominiert D, wird aber selbst von A, B und C dominiert. Im Gegensatz dazu dominieren sich die anderen vier Punkte gegenseitig nicht. Prüft man sie paarweise auf Dominanz, ist jeweils der eine Punkt besser bezüglich einer Zielfunktion und der Zweite besser bezüglich der Anderen. Mit Hilfe der Dominanzrelation lassen sich die nichtdominierten Punkte wie folgt zu einer Menge zusammenfassen.

Definition 1.15 Ein Punkt $\mathbf{x} \in D$
(i) heißt *nicht-dominiert* bezüglich einer Teilmenge $\breve{D} \subseteq D$, wenn gilt: $\nexists \breve{\mathbf{x}} \in \breve{D} : \breve{\mathbf{x}} \prec \mathbf{x}$
(ii) heißt *paretooptimal*, wenn $\mathbf{x}$ nicht-dominiert bezüglich D ist.
Eine Menge $P \subseteq D$ ist die *Pareto-Menge*, wenn $\forall \mathbf{x} \in P$: $\mathbf{x}$ ist paretooptimal.

Die Zusammenhänge zwischen Dominanz und Pareto-Optimalität sollen anhand des folgenden Beispiels verdeutlicht werden.

Beispiel 1.16 Zweikriterielles Optimierungsproblem

$$\begin{aligned} f_1(x_1, x_2) &= (x_1 - 3)^2 + x_2 \to min \\ f_2(x_1, x_2) &= (x_2 - 4)^2 - 2x_1 \to min \\ 0 &< x_1 < 10 \\ 0 &< x_2 < 10 \end{aligned}$$

Im Gegensatz zum allgemeinen Fall besteht bei diesem einfachen Beispiel die Möglichkeit, eine analytische Lösung des Optimierungsproblems anzugeben.

$$x_2 = \frac{25 - 8x_1}{6 - 2x_1}, \quad x_1 \in \left[\frac{25}{8}, 10\right]$$

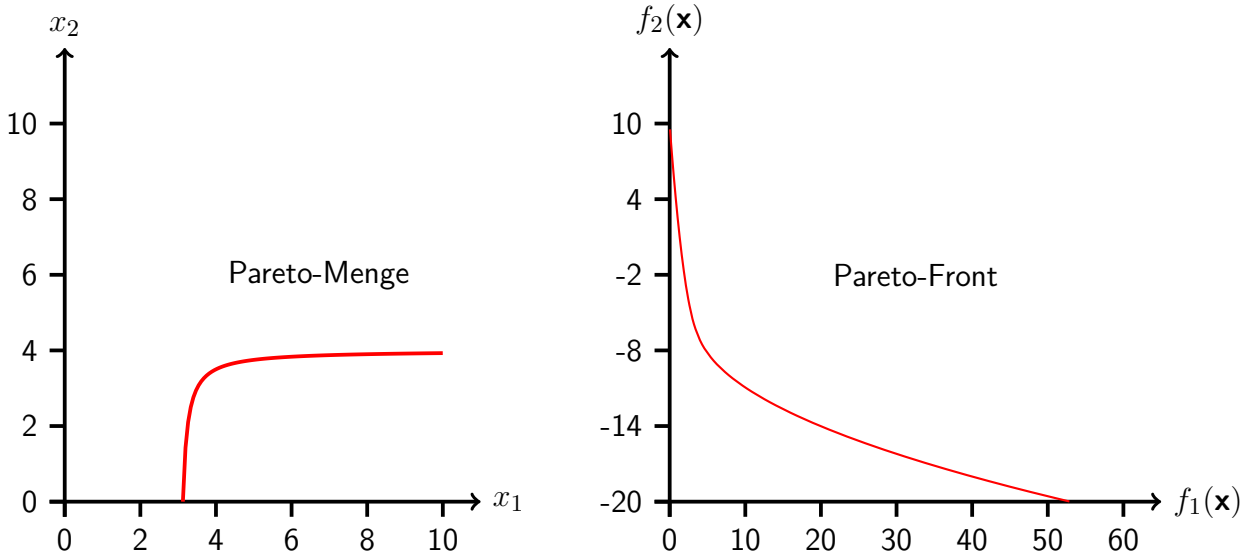


Abbildung 1.4: Lösung im Parameterbereich (links) und im Funktionswertebereich (rechts)

Das linke Bild in Abbildung 1.4 zeigt die Pareto-Menge dieses Optimierungsproblems. Sie besteht aus allen Punkten $\mathbf{x} = (x_1, x_2)$, die innerhalb des Definitionsbereiches nicht dominiert werden. Im rechten Bild wurden die paretooptimalen Punkte in den Zielfunktionsbereich abgebildet. Das Abbild der Pareto-Menge im Funktionswertebereich wird als *Pareto-Front* bezeichnet.

Die Pareto-Front kann je nach Optimierungsaufgabe unterschiedliche Formen annehmen und muss im Allgemeinen nicht zusammenhängend sein.

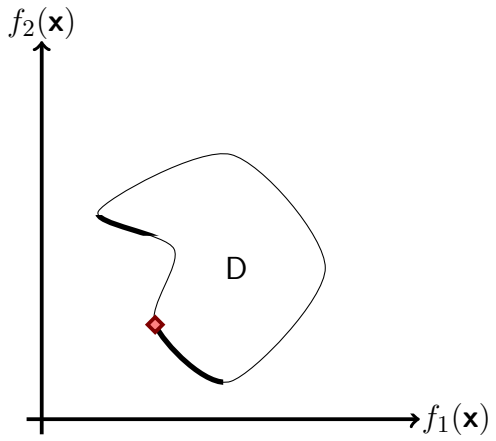


Abbildung 1.5: Zweiteilige Pareto-Front

Nicht zusammenhängende Pareto-Fronten können bei nicht konvexen Optimierungsproblemen auftreten. Das in Abbildung 1.5 dünn dargestellte Stück zwischen den beiden Teilen der Pareto-Front wird vom rot gekennzeichneten Punkt dominiert. Es gehört somit nicht zur Pareto-Front.

Das hier zweidimensional veranschaulichte Konzept der Pareto-Front lässt sich analog auf höherdimensionale Probleme übertragen. Im Allgemeinen führt eine k-kriterielle Optimierungsaufgabe über einem mindestens k-dimensionalen Parameterraum zu einer $(k-1)$-dimensionalen Pareto-Front.

Um multikriterielle Probleme mit einkriteriellen Optimierungsverfahren zu lösen, wurden verschiedene Konzepte entwickelt, die das multikriterielle Problem auf ein einkriterielles zurückführen. Mit diesem Ersatzproblem kann jedoch nicht die gesamte Pareto-Front approximiert werden, da man als Lösung des Ersatzproblems jeweils nur einen Punkt der Pareto-Front erhält. Unter gewissen Umständen ist es sogar möglich, dass bestimmte Bereiche der Pareto-Front niemals Lösung des Ersatzproblems sind.

Ein in der Literatur häufig zu findendes Konzept stellt die *Gewichtete-Summen-Methode* (vgl.[Stö07]) dar. Hierbei werden die einzelnen Zielfunktionen zunächst normiert und anschließend mit einer vom Anwender festgelegten Wichtung aufsummiert. Das so entstandene Ersatzproblem hat die Form

$$\begin{aligned} f(\mathbf{x}) = \sum_{k=1}^{K} w_k f_k(\mathbf{x}) \to min, \quad w_k \in [0,1], \quad \sum_{k=1}^{K} w_k = 1 \\ g_l(\mathbf{x}) = 0, \quad l = 1,...,L \\ h_m(\mathbf{x}) \leq 0, \quad m = 1,...,M \\ \mathbf{x} \in \mathbb{R}^n \end{aligned} \tag{1.7}$$

Die erwähnte Normierung der Zielfunktionen erweist sich in praktischen Anwendungen als schwierig, da dazu die Funktionswertebereiche der einzelnen Ziele bekannt sein müssten. Testrechnungen zur Ermittlung der Wertebereiche sich meist sehr aufwändig.

Das so entstandene einkriterielle Ersatzproblem (1.7) kann mit den gängigen Methoden der einkriteriellen Optimierung gelöst werden. Als Optimallösung ergibt sich, je nach Wahl der Gewichte, ein Punkt der Pareto-Front.

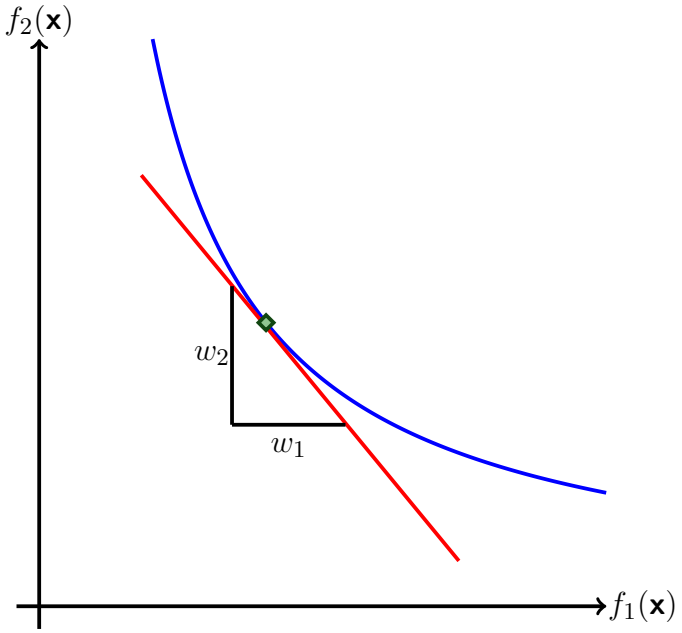


Abbildung 1.6: Gewichtete Summe

Die Abbildung stellt die Arbeitsweise der Gewichteten-Summen-Methode dar. Dabei markiert die blaue Kurve die eigentliche Pareto-Front der multikriteriellen Aufgabe. Der Anstieg der roten Tangente wird durch das Verhältnis der Gewichte bestimmt. Der grün markierte Punkt zeigt die Optimallösung des Ersatzproblems bei dieser konkreten Wahl der Gewichte. Eine Vergrößerung des Gewichts w_1 würde eine Verschiebung des Lösungspunktes in Richtung des Optimums von $f_1(\mathbf{x})$ bewirken.

Ein weiteres Konzept zur Rückführung multikriterieller Probleme auf Einkriterielle ist die *Epsilon-Beschränkungsmethode* (vgl.[Stö07]). Bei diesem Verfahren wird eines der Ziele zur Zielfunktion des Ersatzproblems erklärt. Die anderen Ziele werden in Ungleichheitsnebenbedingungen umgewandelt und durch vom Anwender gewählte Konstanten ϵ_k nach oben beschränkt. Die Wahl dieser Schranken gestaltet sich im Allgemeinen schwierig, da meist nur wenig über die zu erwartende Lösung bekannt ist. Zu groß gewählte Schranken sind wirkungslos, da sie im Bereich der Pareto-Front keine Einschränkung darstellen. Zu klein gewählte Schranken führen möglicherweise zu einem leeren zulässigen Bereich.

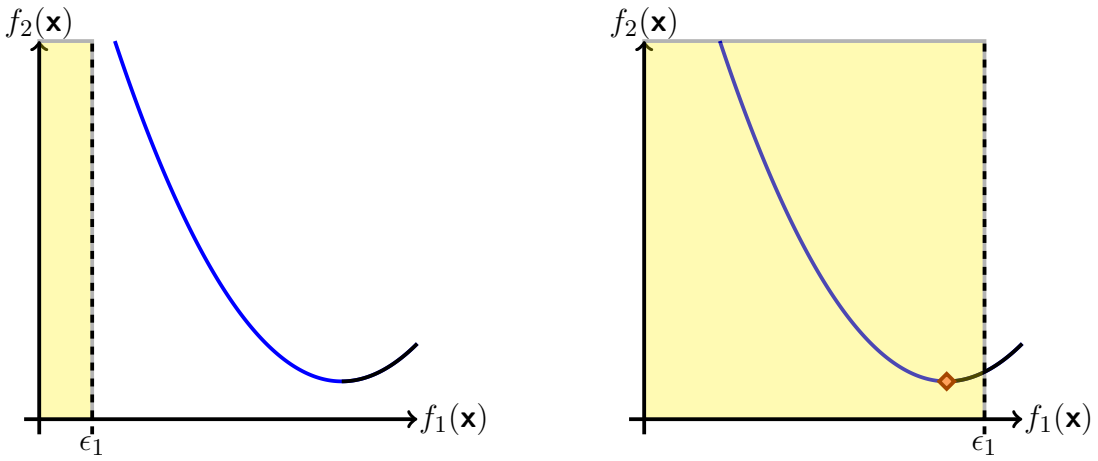


Abbildung 1.7: Schlecht gewählte ϵ Werte

In Abbildung 1.7 wurde $f_2(\mathbf{x})$ zur Ersatzzielfunktion erklärt und $f_1(\mathbf{x})$ durch ϵ_1 beschränkt. Im linken Bild wurde $f_1(\mathbf{x})$ zu stark beschränkt. Die gesamte Pareto-Front liegt damit außerhalb des gelb gekennzeichneten zulässigen Bereiches. Im rechten Bild wurde ϵ_1 zu groß gewählt. Damit hat die Schranke keinen Einfluss auf die Optimierung und die Iteration konvergiert im Minimum bezüglich $f_2(\mathbf{x})$.

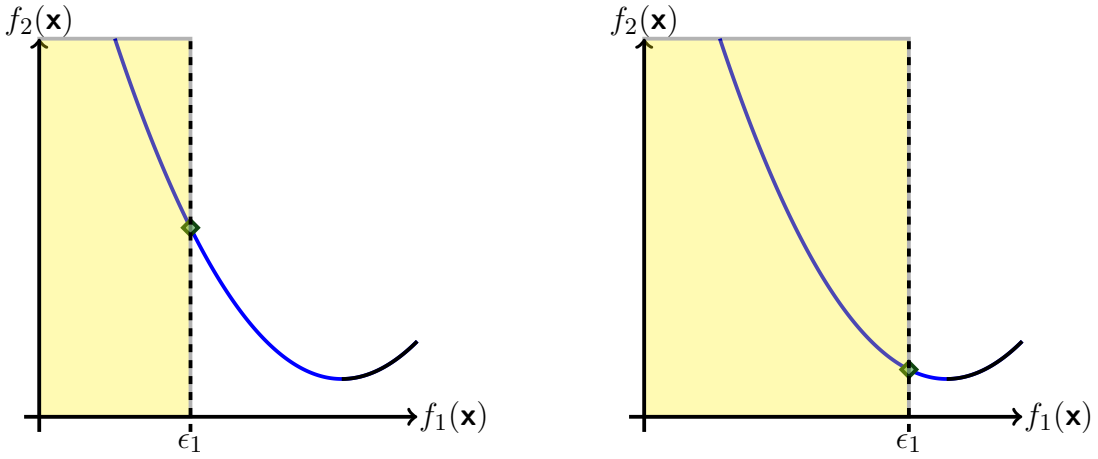


Abbildung 1.8: Gut gewählte ϵ Werte

Bei gut gewählten ϵ-Werten wie in Abbildung 1.8 erhält man jeweils einen Punkt der Pareto-Front als Ergebnis der Optimierung. Eine Approximation der Pareto-Front ist durch mehrere Optimierungen mit leicht veränderten ϵ-Werten möglich.

1.3 Überblick über Klassen von Optimierungsverfahren

Neben gradientenbasierten Verfahren gibt es im Bereich der nichtlinearen Optimierung noch eine Vielzahl anderer Methoden, die mit verschiedensten Ansätzen versuchen, Lösungen der Optimierungsprobleme zu finden. Zu den wichtigsten Vertretern zählen hierbei genetische Algorithmen und Evolutionsstrategien. Diese versuchen, den natürlichen Prozess der Evolution nachzuempfinden und arbeiten nach den Prinzip "Überleben der Stärksten".
Ein anderes häufig angewendetes Verfahren ist das Simulated Annealing (Simuliertes Abkühlen). Grundlage dafür ist die Suche nach einem Zustand minimaler Energie, der sich durch eine sehr langsame Abkühlung einstellt.
Wieder andere Algorithmen orientieren sich an Vorgängen in Ameisenkolonien oder Fischschwärmen. Die folgende Grafik soll einen groben Überblick über die verschiedenen Klassen von Optimierungsalgorithmen geben.

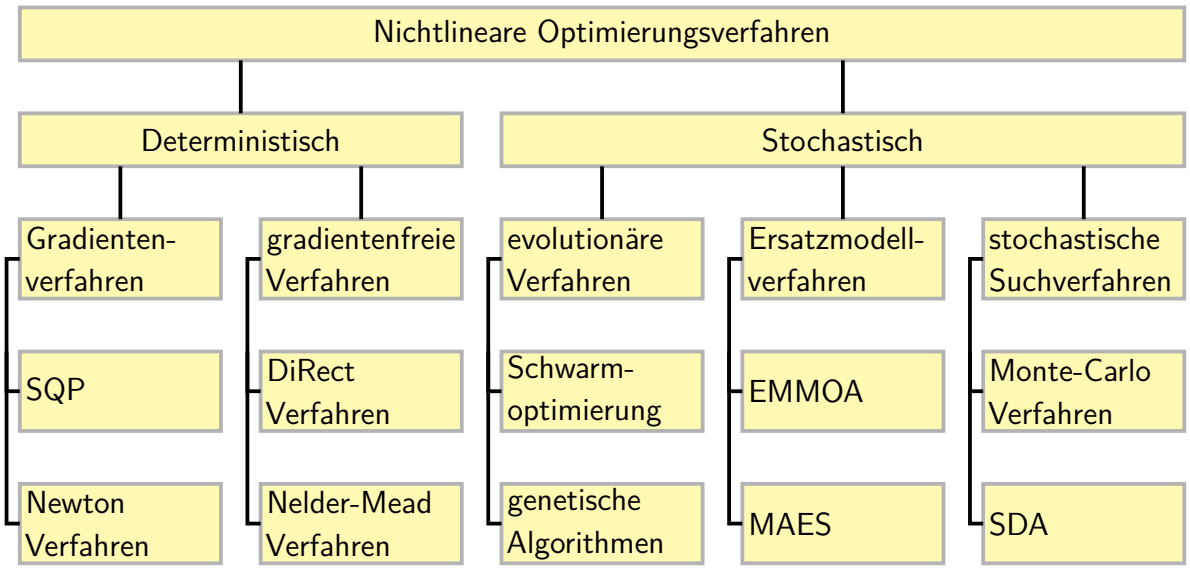


Abbildung 1.9: Klassifizierung nichtlinearer Optimierungsverfahren

2 Gradientenbasierte Verfahren zur einkriteriellen Optimierung

Gradientenbasierte Verfahren zählen zu den deterministischen Optimierungsmethoden und berechnen für ein Optimierungsproblem $f(\mathbf{x}) \rightarrow min$ bei gleichem Startpunkt $\mathbf{x}_0$ stets die gleiche Lösung. Sie zeichnen sich durch gute Konvergenzeigenschaften und hohe Genauigkeit beim Auffinden von Minima eines Optimierungsproblems aus. Aufgrund dieser Eigenschaften eignen sich sehr gut für technische Anwendungen wie z.B. Getriebeoptimierungen.

Gradientenbasierte Verfahren berechnen, ausgehend vom aktuellen Iterationspunkt, eine Suchrichtung mit Hilfe von Gradienteninformationen. Anschließend wird für die so ermittelte Richtung eine Schrittweite bestimmt, die den Zielfunktionswert hinreichend reduziert. Falls die von den Verfahren benötigten Gradienteninformationen nicht analytisch verfügbar sind, greift man auf numerische Methoden zur Bestimmung der Ableitung bzw. des Gradienten zurück. Diese approximieren die Ableitung durch einen Differenzenquotienten.

Unter den Gradientenverfahren zählen neben Sequential Quadratic Programming (SQP) die sogenannten Quasi-Newton-Verfahren und dabei speziell das in dieser Arbeit betrachtete BFGS-Verfahren (nach **B**royden-**F**letcher-**G**oldfarb-**S**hanno) zu den Bekanntesten. Es wurde 1970 erstmals beschrieben und gilt bis heute als Bestes seiner Klasse. Die Grundlage dazu bildet das Newton-Verfahren, welches einführend vorgestellt wird.

2.1 Newton-Verfahren

Das Newton-Verfahren (vgl.[Sch00]) ist eine iterative Methode zur Lösung eines nichtlinearen Gleichungssysteme $F(\mathbf{x}) = 0$. Hierbei ist

$$F(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_K(\mathbf{x}) \end{pmatrix}.$$

Approximiert man das Gleichungssystem durch eine Taylorentwicklung am Punkt $\mathbf{x}_0$ bis zum linearen Glied erhält man

$$F(\mathbf{x}) \approx F(\mathbf{x}_0) + J_F(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) = 0,$$

wobei $J_F(\mathbf{x}_0) = F'(\mathbf{x}_0)$ die Jacobi-Matrix des Systems bezeichnet. Löst man diese Gleichung nun nach $\mathbf{x}$ auf, ergibt sich

$$\begin{aligned} J_F(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) &= -F(\mathbf{x}_0) \\ (\mathbf{x} - \mathbf{x}_0) &= -J_F^{-1}(\mathbf{x}_0)F(\mathbf{x}_0) \\ \mathbf{x} &= \mathbf{x}_0 - J_F^{-1}(\mathbf{x}_0)F(\mathbf{x}_0) \end{aligned} \tag{2.1}$$

Der so berechnete Punkt $\mathbf{x}$ wird als Ausgangspunkt des nächsten Iterationsschrittes verwendet. Die Iteration wird abgebrochen, wenn die Norm der aktuellen Korrektur $J_F^{-1}(\mathbf{x}_k)F(\mathbf{x}_k)$ kleiner als ein festgelegter Wert $\epsilon > 0$ ist.

Bemerkung 2.1 In der Praxis wird es grundsätzlich vermieden, Matrizen zu invertieren. Stattdessen wird das Problem auf die Lösung eines Gleichungssystems zurückgeführt.

Es ergibt sich der folgende Algorithmus:

1: wähle Startpunkt $\mathbf{x}_0 \in \mathbb{R}^n$, setze $k = 0$
2: **repeat**
3: Bestimme $F(\mathbf{x}_k)$ und $J_F(\mathbf{x}_k)$
4: Berechne $\mathbf{d}_k$ als Lösung des linearen Gleichungssystems $J_F(\mathbf{x}_k)\mathbf{d}_k = -F(\mathbf{x}_k)$
5: Berechne $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$
6: Setze $k = k + 1$
7: **until** $\|\mathbf{d}_{k-1}\| < \epsilon$
8: **return** $\mathbf{x}_k$

Algorithmus 1: Newton-Verfahren

Das Newton-Verfahren reduziert die Lösung eines nichtlinearen Gleichungssystems auf das mehrfache Lösen eines linearen Gleichungssystems. Für den Spezialfall, dass $F(\mathbf{x})$ aus nur einer Gleichung in einer Variablen besteht, entspricht das Newton-Verfahren der iterativen Suche nach einer Nullstelle. Das folgende Beispiel illustriert diesen Sachverhalt.

Beispiel 2.2 $f(x) = \frac{x^3}{40} - \frac{3x^2}{8} + \frac{13x}{20} + 3$ mit Lösung $f(5) = 0$

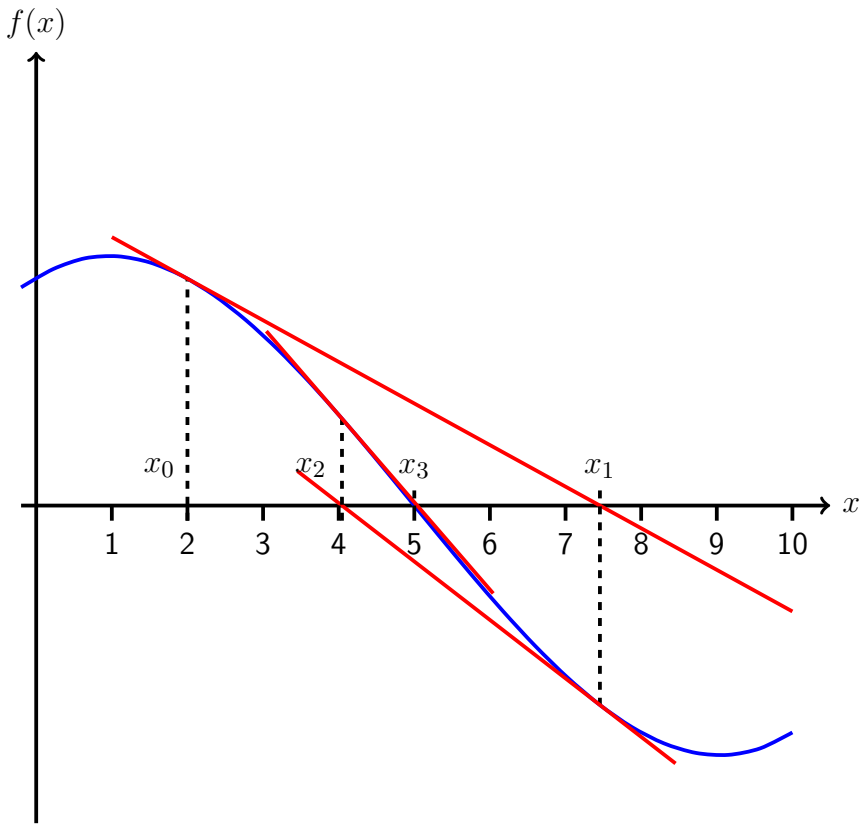


Abbildung 2.1: Prinzip des Newton-Verfahrens

Die Abbildung 2.1 zeigt die Arbeitsweise des Newton-Verfahrens. Ausgehend vom Startpunkt x_0 nähert man sich der Nullstelle der Funktion $f(x)$ durch wiederholte lineare Approximation an.

Schritt	Näherung
0	2.000000000000
1	7.454545454545
2	4.043632476556
3	5.037821225505
4	4.999997791594
5	5.000000000221
6	5.000000000000

Tabelle 2.1: Iterationen für das Beispiel

Konvergenz

Das Newton-Verfahren zählt zu den lokal konvergenten Verfahren, das heißt, es konvergiert sicher gegen eine Lösung $\mathbf{x}^*$, wenn sich der Startpunkt $\mathbf{x}_0$ schon "hinreichend nahe" an der Lösung befindet. Ist dies nicht der Fall, kann unter Umständen Divergenz oder Oszillation auftreten. Innerhalb des Konvergenzbereiches ist das Verfahren quadratisch konvergent und damit gilt in jedem Schritt

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq c\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 \quad , c > 0$$

Das quadratische Konvergenzverhalten zeigt sich in Tabelle 2.1. Ab der dritten Iteration erhöht sich die Genauigkeit um mehrere Nachkommastellen pro Schritt.

Bei der Anwendung auf Optimierungsprobleme wird das Gleichungssystem (1.5) der notwendigen Bedingung für einen stationären Punkt mit den Newton-Verfahren gelöst. Die folgende Herleitung beschränkt sich auf unrestringierte Optimierungsprobleme.

Gesucht ist die Lösung des Gleichungssystems

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} = 0.$$

Dazu wird der Formalismus (2.1) auf die Funktion $\nabla f(\mathbf{x})$ angewendet. Die Taylorentwicklung von $\nabla f(\mathbf{x})$ wird nach dem linearen Glied abgebrochen.

$$\begin{aligned} \nabla f(\mathbf{x}) &= \nabla f(\mathbf{x}_0) + \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \dots \\ \nabla f(\mathbf{x}) &\approx \nabla f(\mathbf{x}_0) + \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \end{aligned}$$

Die so erhaltene Gleichung kann nun durch Null setzen und Umstellung nach $\mathbf{x}$ aufgelöst werden.

$$\begin{aligned} 0 &= \nabla f(\mathbf{x}_0) + \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \\ -\nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) &= \nabla f(\mathbf{x}_0) \\ \mathbf{x} - \mathbf{x}_0 &= -(\nabla^2 f(\mathbf{x}_0))^{-1} \nabla f(\mathbf{x}_0) \\ \mathbf{x} &= \mathbf{x}_0 - (\nabla^2 f(\mathbf{x}_0))^{-1} \nabla f(\mathbf{x}_0) \end{aligned} \tag{2.2}$$

Ein Vergleich der Darstellungen (2.1) und (2.2) zeigt, dass die Hesse-Matrix gleichzeitig die Jacobi-Matrix des Gradienten ist.

Im Weiteren wird $-(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$ als Korrekturrichtung $\mathbf{d}_k \in \mathbb{R}^n$ bezeichnet. Der Newton-Schritt ergibt sich zu

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k.$$

Das so erhaltene Verfahren wird als ungedämpftes Newton-Verfahren bezeichnet. Dieses ist, da es direkt aus dem eindimensionalen Newton-Verfahren abgeleitet wurde, wieder lokal quadratisch konvergent und kann durch eine Schrittweitensteuerung mit Parameter $\alpha > 0$ (genauer erklärt in 2.2.2) globalisiert werden.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}_k, \quad \alpha \in \mathbb{R}, \quad \alpha \in (0,1] \tag{2.3}$$

Gleichung (2.3) wird als gedämpftes Newton-Verfahren bezeichnet.

Durch die Globalisierung verliert das Verfahren seine quadratische Konvergenz. Man kann jedoch zeigen, dass das gedämpfte Newton-Verfahren immer noch superlinear konvergiert.
Ein Nachteil des Newton-Verfahrens ist, dass die Hesse-Matrix $\nabla^2 f(\mathbf{x})$ im Allgemeinen nicht immer positiv definit ist. Diese Eigenschaft ist jedoch wichtig, um zu sichern, dass $\mathbf{d}_k$ eine Abstiegsrichtung ist.

Definition 2.3 Sei $f : D \to \mathbb{R}$ eine stetig differenzierbare Funktion mit $D \subseteq \mathbb{R}^n$. Ein Vektor $\mathbf{d} \in \mathbb{R}^n$ heißt *Abstiegsrichtung* von f im Punkt $\mathbf{x} \in \mathbb{R}^n$, wenn

$$\nabla f(\mathbf{x})^T \mathbf{d} < 0$$

gilt.

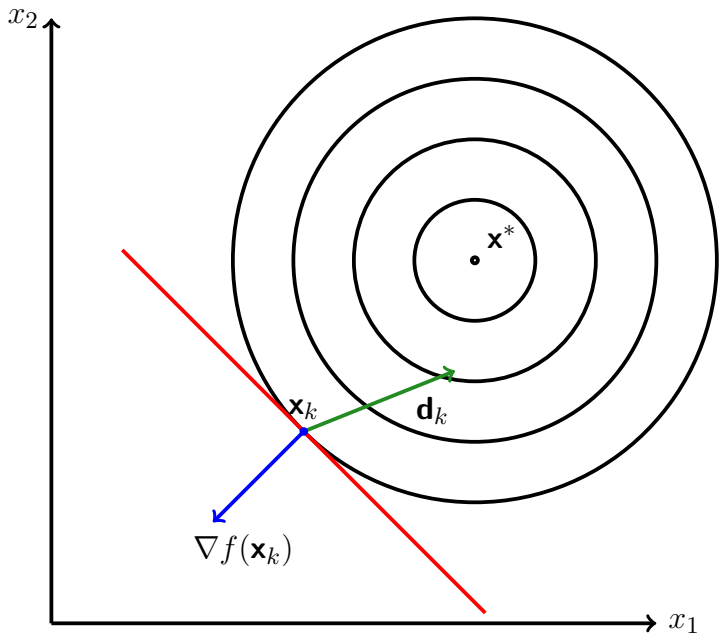


Abbildung 2.2: Mögliche Abstiegsrichtung $\mathbf{d}_k$

Die Abbildung 2.2 zeigt die Höhenlinien einer Zielfunktion (Kreise). Der Gradient (blau) steht an jedem Punkt $\mathbf{x}_k$ senkrecht auf der Tangente (rot) der Höhenlinie. Damit ist lokal jeder Vektor, der mit dem Gradienten eine Winkel $> 90°$ einschließt, eine Abstiegsrichtung und somit folgt die Bedingung für das Skalarprodukt $\nabla f(\mathbf{x}_k)^T \mathbf{d}_k < 0$.

Für die Newton-Richtung $\mathbf{d}_k$ ergibt sich

$$\begin{aligned} \mathbf{d}_k &= -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \\ \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k &= -\nabla f(\mathbf{x}_k) \\ \mathbf{d}_k^T \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k &= -\mathbf{d}_k^T \nabla f(\mathbf{x}_k) = -\nabla f(\mathbf{x}_k)^T \mathbf{d}_k > 0 \end{aligned}$$

Die Forderung $\mathbf{d}_k^T \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k > 0$ wird in jedem Fall erfüllt, wenn die Hesse-Matrix positiv definit ist. Damit ist gesichert, dass $\mathbf{d}_k$ eine Abstiegsrichtung ist.

2.2 Quasi-Newton-Verfahren

Ein weiterer Nachteil des Newton-Verfahrens ist der hohe Aufwand zur Berechnung der exakten Hesse-Matrix. Um dies zu umgehen, wurde auf der Basis des Newton-Verfahrens die Klasse der sogenannten Quasi-Newton-Verfahren entwickelt. Dabei wird versucht, die Nachteile des Newton-Verfahrens zu vermeiden und die Vorteile zu bewahren. Ziel dieser Verfahrensklasse ist es, statt der exakten Hesse-Matrix $\nabla^2 f(\mathbf{x}_k)$ eine Approximation H_k zu verwenden, die in jedem Schritt ähnlich wie der Vektor $\mathbf{x}_{k+1}$ aus der vorherigen Approximation und einem additiven Update berechnet wird. Es soll also

$$H_{k+1} = H_k + M_k$$

berechnet werden. Das Update M_k wird aus den an die Matrix H_{k+1} gestellten Bedingungen bestimmt. Durch geschickte Wahl der Bedingungen ist es möglich, die Symmetrie sowie die positive Definitheit der Matrix H_{k+1} für eine positiv definite Matrix H_k zu sichern und das Update mit vergleichsweise geringem Aufwand zu brechnen.

2.2.1 BFGS-Verfahren

Die verschiedenen Quasi-Newton-Verfahren unterscheiden sich nur in der Update-Vorschrift für die Approximation der Hesse-Matrix. In der praktischen Anwendung stellt sich das BFGS-Verfahren (vgl. [JS00],[Obe12]) als das allgemein Beste seiner Art heraus. Daher soll es im Folgenden aus den an die Matrix H_{k+1} gestellten Bedingungen hergeleitet werden. Wie beim Newton-Verfahren beschränkt sich die Einführung vorerst auf unrestringierte Probleme. Anschließend wird eine Möglichkeit aufgezeigt, Nebenbedingungen in die Berechnung einzubinden.

Eine erste Bedingung ergibt sich aus der Taylor-Entwicklung in (2.2) durch Einsetzen von x_{k+1}

$$\begin{aligned}\nabla f(\mathbf{x}_k) &= \nabla f(\mathbf{x}_{k+1}) + \nabla^2 f(\mathbf{x}_{k+1})(\mathbf{x}_k - \mathbf{x}_{k+1}) \\ \nabla^2 f(\mathbf{x}_{k+1})(\mathbf{x}_{k+1} - \mathbf{x}_k) &= \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \\ H_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) &= \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\end{aligned}$$

Im Folgenden bezeichnen

$$\begin{aligned}\mathbf{d}_k &:= \mathbf{x}_{k+1} - \mathbf{x}_k \\ \mathbf{y}_k &:= \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\end{aligned}$$

und es ergibt sich

$$H_{k+1}\mathbf{d}_k = \mathbf{y}_k. \tag{2.4}$$

Gleichung (2.4) wird als Quasi-Newton-Gleichung bezeichnet. H_{k+1} soll so konstruiert werden, dass die Quasi-Newton-Gleichung erfüllt ist.

Definition 2.4 Ein Verfahren, bei dem H_{k+1} aus H_k, $\mathbf{y}_k$ und $\mathbf{d}_k$ hervorgeht und welches die Quasi-Newton-Gleichung erfüllt, heißt Quasi-Newton-Verfahren.

Da die Bestimmung von H_{k+1} durch die Quasi-Newton-Gleichung nicht eindeutig ist, wurden verschiedene Quasi-Newton-Updates entwickelt. Ein einfaches Update, welches die Symmetrie der Matrix erhält, ergibt sich aus dem Ansatz

$$H_{k+1} = H_k + \alpha \mathbf{u}\mathbf{u}^T \quad , \alpha \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^n. \tag{2.5}$$

Eingesetzt in die Quasi-Newton Bedingung ergibt sich

$$H_k\mathbf{d}_k + \alpha \mathbf{u}(\mathbf{u}^T\mathbf{d}_k) = \mathbf{y}_k.$$

Setzt man in dieser Gleichung

$$\alpha = \frac{1}{\mathbf{u}^T\mathbf{d}_k}, \qquad \mathbf{u} = \mathbf{y}_k - H_k\mathbf{d}_k,$$

erhält man ein Update der Form

$$H_{k+1} = H_k + \frac{(\mathbf{y}_k - H_k\mathbf{d}_k)(\mathbf{y}_k - H_k\mathbf{d}_k)^T}{(\mathbf{y}_k - H_k\mathbf{d}_k)^T\mathbf{d}_k}.$$

Die so erhaltene Gleichung wird als symmetrisches Rang 1 Update bezeichnet. Dieses garantiert die Symmetrie, im Allgemeinen jedoch nicht die positive Definitheit von H_{k+1}. Ein weiterer Nachteil besteht darin, das der Nenner unter gewissen Umständen Null werden kann. Ein Rang 1 Update reicht somit nicht aus, um alle gestellten Foderungen

zu erfüllen. Daher erweitert man den Ansatz um ein zweites Rang 1 Update.

$$H_{k+1} = H_k + \alpha \mathbf{u}\mathbf{u}^T + \beta \mathbf{v}\mathbf{v}^T \quad , \alpha, \beta \in \mathbb{R}, \quad \mathbf{u}, \mathbf{v} \in \mathbb{R}^n.$$

Setzt man für das erste Update

$$\mathbf{u} := H_k \mathbf{d}_k, \qquad \alpha = \frac{-1}{\mathbf{d}_k^T H_k \mathbf{d}_k},$$

ergibt sich daraus der Zwischenschritt

$$\widetilde{H}_k = H_k - \frac{(H_k \mathbf{d}_k)(H_k \mathbf{d}_k)^T}{\mathbf{d}_k^T H_k \mathbf{d}_k}.$$

Diese Wahl von $\mathbf{u}$ und α sichert, dass $\widetilde{H}_k$ zumindest positiv semidefinit ist.

Beweis: Für $\widetilde{H}_k$ und $\mathbf{x} \in \mathbb{R}^n$ gilt

$$\mathbf{x}^T \widetilde{H}_k \mathbf{x} = \mathbf{x}^T H_k \mathbf{x} - \frac{\mathbf{x}^T H_k \mathbf{d}_k \mathbf{d}_k^T H_k \mathbf{x}}{\mathbf{d}_k^T H_k \mathbf{d}_k}$$

$$\mathbf{d}_k^T H_k \mathbf{d}_k \mathbf{x}^T \widetilde{H}_k \mathbf{x} = \mathbf{d}_k^T H_k \mathbf{d}_k \mathbf{x}^T H_k \mathbf{x} - \mathbf{x}^T H_k \mathbf{d}_k \mathbf{d}_k^T H_k \mathbf{x}$$

$$= \langle \mathbf{d}_k, H_k \mathbf{d}_k \rangle \langle \mathbf{x}, H_k \mathbf{x} \rangle - \langle \mathbf{x}, H_k \mathbf{d}_k \rangle^2$$

$$\underbrace{\mathbf{d}_k^T H_k \mathbf{d}_k}_{\geq 0} \mathbf{x}^T \widetilde{H}_k \mathbf{x} = \underbrace{\langle \mathbf{d}_k, \mathbf{d}_k \rangle_{H_k} \langle \mathbf{x}, \mathbf{x} \rangle_{H_k} - \langle \mathbf{x}, \mathbf{d}_k \rangle_{H_k}^2}_{Cauchy-Schwarzsche-Ungleichung} \geq 0,$$

wobei $\langle \mathbf{x}, \mathbf{x} \rangle_{H_k}$ für das von der positiv definiten, symmetrischen Matrix H_k induzierte Skalarprodukt $\langle \mathbf{x}, H_k \mathbf{x} \rangle$ steht. □

Das zweite Update ergibt sich nun aus der Forderung, dass die Quasi-Newton-Gleichung erfüllt werden soll. Durch Einsetzen des Zwischenschrittes mit dem noch fehlenden zweiten Update

$$H_{k+1} = H_k - \frac{(H_k \mathbf{d}_k)(H_k \mathbf{d}_k)^T}{\mathbf{d}_k^T H_k \mathbf{d}_k} + \beta \mathbf{v}\mathbf{v}^T \quad , \beta \in \mathbb{R}, \quad \mathbf{v} \in \mathbb{R}^n$$

in Gleichung (2.4) erhält man

$$H_k \mathbf{d}_k - \frac{(H_k \mathbf{d}_k)(H_k \mathbf{d}_k)^T \mathbf{d}_k}{\mathbf{d}_k^T H_k \mathbf{d}_k} + \beta \mathbf{v}(\mathbf{v}^T \mathbf{d}_k) = \mathbf{y}_k$$

$$\beta \mathbf{v}(\mathbf{v}^T \mathbf{d}_k) = \mathbf{y}_k$$

Mit der Wahl von $\mathbf{v} := \mathbf{y}_k$ und $\beta = \frac{1}{\mathbf{y}_k^T \mathbf{d}_k}$ ist auch diese Gleichung erfüllt. Zusammengefasst ergibt sich aus den beiden Updates schließlich die BFGS-Formel

$$H_{k+1} = H_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{d}_k} - \frac{(H_k \mathbf{d}_k)(H_k \mathbf{d}_k)^T}{\mathbf{d}_k^T H_k \mathbf{d}_k}. \tag{2.6}$$

Man kann zeigen (vgl. [Alt02]), dass für das BFGS-Update unter der Voraussetzung $\mathbf{y}^T \mathbf{d} > 0$ gilt

$$H_k \, positiv \, definit \Rightarrow H_{k+1} \, positiv \, definit.$$

Diese Voraussetzung ist allgemein nicht gegeben, kann aber durch eine Schrittweitensteuerung, wie in 2.2.2 beschrieben, erzwungen werden.

Die Bestimmung der Suchrichtung erfordert in jedem Schritt das Lösen des Gleichungssystems $H_k \mathbf{d}_k = -\nabla f(\mathbf{x}_k)$. Um dies zu vermeiden, wird häufig das inverse BFGS-Verfahren angewendet. Statt der Approximation H_k verwendet man hier eine Approximation der inversen Hesse-Matrix $B_k = H_k^{-1}$. Dadurch reduziert sich die Berechnung der Suchrichtung auf die Matrizenmultiplikation $\mathbf{d}_k = -B_k \nabla f(\mathbf{x}_k)$. Für B_k ergibt sich nach einigen Umformungen der Gleichung (2.6) das inverse BFGS-Update

$$B_{k+1} = B_k + \frac{(\mathbf{d}_k - B_k \mathbf{y}_k)\mathbf{d}_k^T + \mathbf{d}_k(\mathbf{d}_k - B_k \mathbf{y}_k)^T}{\mathbf{y}_k^T \mathbf{d}_k} - \frac{(\mathbf{d}_k - B_k \mathbf{y}_k)^T \mathbf{y}_k}{(\mathbf{y}_k^T \mathbf{d}_k)^2} \mathbf{d}_k \mathbf{d}_k^T$$

Bemerkung 2.5 Unter Ausnutzung der Symmetrie von H_k wird in [DS87] eine Variante vorgestellt, die mit einem Update der Cholesky-Zerlegung von H arbeitet. Für $H_k = L_k L_k^T$ ist

$$H_{k+1} = JJ^T \; mit \; J = L_k + \frac{(\mathbf{y} - L\mathbf{w})\mathbf{w}^T}{\mathbf{w}^T \mathbf{w}} \; sowie \; \mathbf{w} = \sqrt{\frac{\mathbf{y}^T \mathbf{d}}{\mathbf{d}^T H \mathbf{d}}} L_k^T \mathbf{d}$$

Weiter berechnet man eine QR-Zerlegung von J^T und erhält

$$H_{k+1} = JJ^T = R^T Q^T Q R = R^T R$$

und damit

$$L_{k+1} = R^T.$$

Zur Initialisierung der Matrix H wird häufig die Einheitsmatrix verwendet, da diese sowohl symmetrisch als auch positiv definit ist. Bei dieser Wahl entspricht der erste Iterationsschritt einem Gradientenschritt.

2.2.2 Schrittweitensteuerung

Um die bisher lokal konvergenten Verfahren zu globalisieren, d.h. ein konvergentes Verhalten von jedem beliebigen Startpunkt $\mathbf{x}_0 \in D$ aus zu erreichen, eignet sich eine Schrittweitensteuerung. Diese dient dazu, die Länge des Korrekturschrittes gegebenenfalls einzuschränken, um divergentes Verhalten zu verhindern und einen Abstieg im aktuellen Schritt zu sichern. Unter der Schrittweitenberechnung versteht man die Minimierung der Zielfunktion $f(\mathbf{x}_k)$ entlang der Suchrichtung $\mathbf{d}_k$. Das Problem hat die Form

$$\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{d}_k).$$

Die Berechnung der exakten Schrittweite ist ein eigenes Minimierungsproblem, das in jedem Schritt gelöst werden müsste. Es erweist sich daher als sehr aufwändig, die exakte Schrittweite zu berechnen. Die Verwendung dieser exakten Schrittweite würde allerdings zur maximalen Verbesserung des Zielfunktionswertes im aktuellen Schritt führen. An dieser Stelle muss ein Kompromiss zwischen Berechnungsaufwand und Verbesserung des Zielfunktionswertes gefunden werden.
Der im Folgenden vorgestellte Liniensuchalgorithmus bestimmt durch sukzessive Vergrößerung bzw. Verkleinerung des Parameters α ein Intervall entlang der Suchrichtung, welches ein (zumindest lokales) Minimum enthält. Anschließend wird das Minimum innerhalb des Intervalls mittels Trisektion angenähert.

Vor Beginn der Liniensuche wird geprüft, ob ein ungedämpfter Newton-Schritt ($\alpha = 1$) den Zielfunktionswert reduziert. Wenn ja, wird $\alpha = 1$ als Schrittweite akzeptiert und die Liniensuche abgebrochen. Durch diese Prüfung werden vor allem in den letzten Iterationsschritten nahe des Minimums die guten lokalen Konvergenzeigenschaften ausgenutzt.

Schlägt die Prüfung fehl, wird durch Variation von α ein Intervall bestimmt, welches einen Wert von α enthält, der zu einem Minimum der Zielfunktion entlang der Suchrichtung führt.

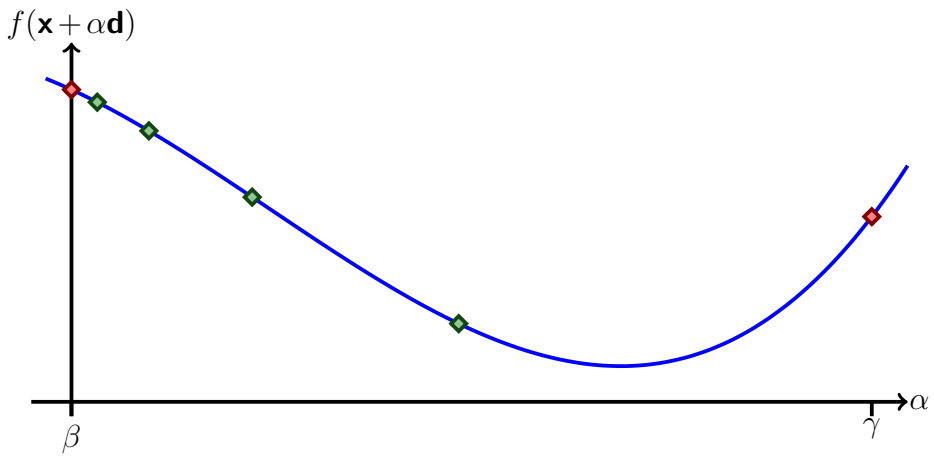


Abbildung 2.3: Sukzessive Vergrößerung von α

Abbildung 2.3 zeigt die sukzessive Verdopplung von α. Diese wird so lange fortgesetzt, bis die Funktionswerte wieder ansteigen. Die beiden roten Endpunkte bilden das Intervall, welches das Minimum enthält.

Das so ermittelte Intervall $[\beta, \gamma]$ wird nun mittels Trisektion schrittweise verkleinert. Trisektion bietet gegenüber Bisektion den Vorteil, dass in jedem Schritt zwei innere Punkte des Intervalls berechnet werden, die bezüglich ihrer Zielfunktionswerte verglichen werden können. Dazu werden die Funktionswerte $f\left[\mathbf{x} + \left(\beta + \frac{\gamma - \beta}{3}\right)\mathbf{d}\right]$ sowie $f\left[\mathbf{x} + \left(\beta + \frac{2(\gamma - \beta)}{3}\right)\mathbf{d}\right]$ der beiden inneren Punkte berechnet, um anschließend das Intervall entsprechend dem folgendem Algorithmus zu verkleinern.

1: setze $\beta = 0$, $\gamma = \alpha$ (α wie in Abbildung 2.3 ermittelt),
2: **for** k=1 **to** 4 **do**
3: Berechne $f_{1/3} = f\left[\mathbf{x} + \left(\beta + \frac{\gamma - \beta}{3}\right)\mathbf{d}\right]$, $f_{2/3} = f\left[\mathbf{x} + \left(\beta + \frac{2(\gamma - \beta)}{3}\right)\mathbf{d}\right]$
4: **if** $f_{1/3} < f_{2/3}$ **then**
5: $\gamma = \beta + \frac{2(\gamma - \beta)}{3}$
6: **else**
7: $\beta = \beta + \frac{\gamma - \beta}{3}$
8: **end if**
9: **end for**
10:
11: **if** $f\left[\mathbf{x} + \left(\beta + \frac{\gamma - \beta}{3}\right)\mathbf{d}\right] \leq f\left[\mathbf{x} + \left(\beta + \frac{2(\gamma - \beta)}{3}\right)\mathbf{d}\right]$ **then**
12: **return** β
13: **else**
14: **return** γ
15: **end if**

Algorithmus 2: Trisektion

Die Festlegung auf vier Trisektionsschritte in Zeile zwei des Algorithmus ist heuristisch und hat sich praktisch bewährt.

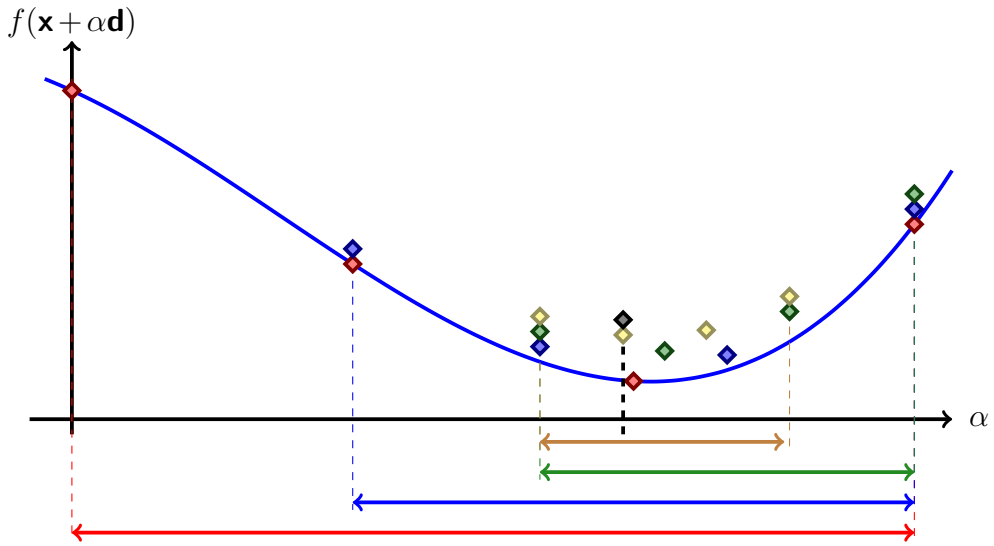


Abbildung 2.4: Vier Trisektionsschritte rot → blau → grün → gelb

Abbildung 2.4 zeigt die Verkleinerung des Intervalls durch Trisektion. In jedem Schritt wird das Intervall in drei Teile unterteilt und der Teil mit den größten Funktionswerten verworfen. Das Intervall verkleinert sich dadurch in k Trisektionsschritten um den Faktor $\left(\frac{2}{3}\right)^k$.

Mit der Schrittweitensteuerung wird das globale BFGS-Verfahren wie folgt formuliert

1: wähle Startpunkt $\mathbf{x}_0 \in \mathbb{R}$, $H_0 = I$, $\epsilon > 0$, setze $k = 0$
2: **repeat**
3: Bestimme $\nabla f(\mathbf{x}_k)$
4: Löse $H_k \mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
5: Berechne Schrittweite α_k nach 2.2.2
6: Berechne Schritt $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
7: Berechne $\mathbf{y}_k := \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k), \quad \mathbf{s}_k = \alpha_k \mathbf{d}_k$
8: **if** $\mathbf{y}^T \mathbf{s} > 0$ **then**
9: Berechne $H_{k+1} = H_k + \dfrac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \dfrac{(H_k \mathbf{s}_k)(H_k \mathbf{s}_k)^T}{\mathbf{s}_k^T H_k \mathbf{s}_k}$
10: **end if**
11: Setze $k = k + 1$
12: **until** $\|\mathbf{d}_k\| < \epsilon$
13: **return** $\mathbf{x}_{k+1}$

Algorithmus 3: BFGS-Verfahren

Konvergenz

In Algorithmus 3 ist durch die positive Definitheit der BFGS-Matrix H_k gesichert, dass $\mathbf{d}_k$ lokal eine Abstiegsrichtung ist und die Schrittweitensteuerung garantiert einen

hinreichenden Abstieg. Damit konvergiert das Verfahren sicher gegen ein Minimum von f, falls f nach unten beschränkt ist. Dabei ist nicht auszuschließen, dass es sich um ein lokales Minimum handelt. Dies liegt aber in der Natur der gradientenbasierten Optimierungsverfahren und kann an dieser Stelle nicht verhindert werden, da die Bedingung aus Satz 1.10, die zum Abbruch der Iteration führt, auch in lokalen Minima erfüllt ist.

2.2.3 Behandlung von Nebenbedingungen

Das BFGS-Verfahren ist wie das Newton-Verfahren für unrestringierte Optimierungsaufgaben konzipiert. Um auch restringierte Probleme mit dem BFGS-Verfahren zu lösen, besteht die Möglichkeit, das restringierte Problem mittels einer Straf- oder Barrierefunktion auf ein unrestringiertes zurückzuführen, welches dann mit dem vorgestellten Algorithmus gelöst werden kann.

Der wesentliche Unterschied zwischen Straf- und Barrierefunktion besteht darin, dass die Straffunktion ein Verlassen des zulässigen Bereiches mit einer Erhöhung des Zielfunktionswertes bestraft, während die Barrierefunktion bereits die Annäherung an den Rand des zulässigen Bereiches mit Strafen belegt.

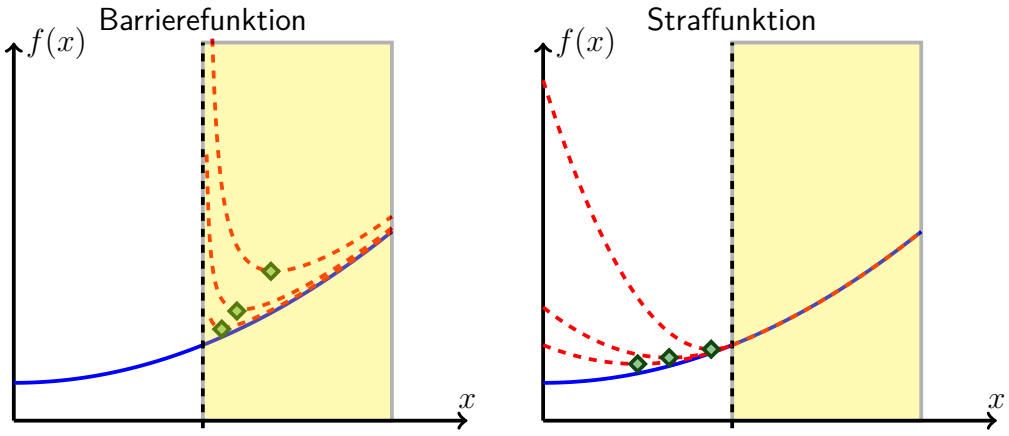


Abbildung 2.5: Prinzip von Straf- und Barrierefunktion

Abbildung 2.5 zeigt die restringierte Optimierung mit Straf- bzw. Barrierefunktion. Der Verlauf der blau dargestellten Zielfunktion ändert sich dabei durch die Addition des Straf- oder Barriereterms (gestrichelte Darstellungen). Im linken Bild werden Annäherung und Überschreitung des Randes des gelb markierten, zulässigen Bereiches mit Strafen belegt. Mit wachsendem Strafparameter nähert sich das Minimum von innen dem Rand des zulässigen Bereiches. Das rechte Bild zeigt die Arbeitsweise einer Straffunktion. Hier nähert sich das Minimum mit wachsendem Strafparameter von außen an den Rand des

zulässigen Bereiches an. Für das BFGS-Verfahren in dieser Arbeit soll eine Straffunktion verwendet werden.

In [JS00] werden verschiedene Straffunktionen beschrieben. Dabei wird das restringierte Problem durch Addition eines Strafterms zur Zielfunktion durch ein unrestringiertes approximiert. Für einen Strafterm $p(\mathbf{x})$ gilt allgemein

$$\begin{aligned} p(\mathbf{x}) &> 0, \quad \mathbf{x} \notin D \\ p(\mathbf{x}) &= 0, \quad \mathbf{x} \in D. \end{aligned}$$

Unzulässige Punkte werden mit einem Zuwachs des Zielfunktionswertes bestraft. Mit dem Strafterm und einem Strafparameter $\sigma > 0$ ergibt sich die allgemeine Penalty-Funktion zu

$$P(\mathbf{x}, \sigma) = f(\mathbf{x}) + \sigma p(\mathbf{x}) \tag{2.7}$$

Für hinreichend große σ sollte die Lösung von $P(\mathbf{x})$ eine gute Näherung für die Lösung von $f(\mathbf{x})$ sein. Ausgehend von der allgemeinen Darstellung (2.7) gibt es eine Vielzahl von Möglichkeiten, eine Penalty-Funktion zu formulieren. Eine Variante, die die für das BFGS-Verfahren notwendige Differenzierbarkeit der Zielfunktion erhält, ist die erweiterte Lagrangefunktion für Gleichungsrestriktionen (vgl.[Har07]).

$$P(\mathbf{x}, \sigma) = L(\mathbf{x}, \lambda, \sigma) = f(\mathbf{x}) - \sum_{l=1}^{L} \lambda_l g_l(\mathbf{x}) + \frac{\sigma}{2} \sum_{l=1}^{L} g_l^2(\mathbf{x}) \tag{2.8}$$

Im Unterschied zur gewöhnlichen Lagrange Funktion wird hier der Strafterm $\frac{\sigma}{2} \sum_{l=1}^{L} g_l^2(\mathbf{x})$ addiert. Das Problem $P(\mathbf{x}, \sigma)$ wird nun mehrfach gelöst, wobei die Lösung eines Verfahrensdurchlaufs stets Startpunkt für den nächsten Durchlauf ist. Nach jedem Lösen von $P(\mathbf{x}, \sigma)$ werden die Lagrange-Multiplikatoren sowie der Strafparameter angepasst.

$$\begin{aligned} \sigma_{k+1} &= c\sigma_k, \quad c > 1 \\ (\lambda_i)_{k+1} &= (\lambda_i)_k - g_l(\mathbf{x}_k)\sigma_k \; \forall l \end{aligned}$$

Mit jeder Iteration vergrößert sich die Strafe, die für die Verletzung von Nebenbedingungen auf die Zielfunktion addiert wird. Der Abbruch erfolgt, wenn alle Restriktionen bis auf ein $\epsilon > 0$ erfüllt sind.

2.2.4 Algorithmische Umsetzung

Das BFGS-Verfahren lag in der hier beschriebenen Version in einer bestehenden IAV Bibliothek vor. Es wurde vom Autor an das in der Optimierungssoftware der "IAV Engineering Toolbox" vorhandene hirarchische Klassensystem angepasst und mittels Testfunktionen auf seine Korrektheit überprüft. Die Bearbeitung nichtlinearer Optimierungsprobleme erfolgt zuverlässig, wenn auch in den meisten Fällen etwas langsamer als beim ebenfalls implementierten SQP Algorithmus.

3 Gradientenbasierte multikriterielle Optimierung

Die multikriterielle Optimierung stellt eine Erweiterung der einkriteriellen Optimierung dar. Ziel der multikriteriellen Optimierungsverfahren ist es, paretooptimale Lösungen wie in Abschnitt 1.2 beschrieben, zu finden. Einführend zu diesem Kapitel soll zunächst auf einige Probleme aufmerksam gemacht werden, die bei der Überführung multikriterieller Probleme in einkriterielle Aufgaben auftreten können. Davon motiviert wird anschließend ein gradientenbasiertes multikriterielles Optimierungsverfahren vorgestellt, welches ohne Ersatzzielfunktionen arbeitet und dadurch die dabei auftretenden Nachteile vermeidet.

Beim Einsatz der Gewichtete-Summen-Methode oder der Epsilon-Beschränkungsmethode kann es dazu kommen, dass die ermittelte Abstiegsrichtung der Ersatzzielfunktion nicht zu einer Verbesserung bezüglich aller Einzelziele führt.

Beispiel 3.1 Gewichtete Summe dreier Zielfunktionen $f_1(x) = sin(x)$, $f_2(x) = cos(x)$, $f_3(x) = (x-\pi)^2$ mit den Gewichten $w_1 = w_2 = w_3 = \frac{1}{3}$ und den Iterationspunkten $x_0 = 0$, $x_1 = \frac{\pi}{2}$

Der Iterationsschritt stellt für die Ersatzzielfunktion ein Verbesserung dar.

$$\sum_{k=1}^{3} w_k f_k(x_1) = \frac{\pi^2+4}{12} \approx 1,16 < \sum_{k=1}^{3} w_k f_k(x_0) = \frac{\pi^2+1}{3} \approx 3,62$$

Betrachtet man jedoch die einzelnen Zielfunktionen separat, wird nur für zwei der drei Ziele eine Verbesserung erzielt.

$$f_1(x_1) > f_1(x_0)$$
$$f_2(x_1) < f_2(x_0)$$
$$f_3(x_1) < f_3(x_0)$$

Trotz der Verbesserung bezüglich der Ersatzzielfunktion hat sich eines der Einzelziele verschlechtert. Derartige Effekte können die Konvergenzgeschwindigkeit negativ beeinflussen, sind jedoch bei Konzepten mit Ersatzzielfunktion unvermeidbar.

Ein weiteres Problem entsteht bei nichtkonvexen Paretofronten. Mit der Gewichtete-Summen-Methode ist es nicht möglich, Punkte auf konkaven Bereichen der Pareto-Front zu erreichen.

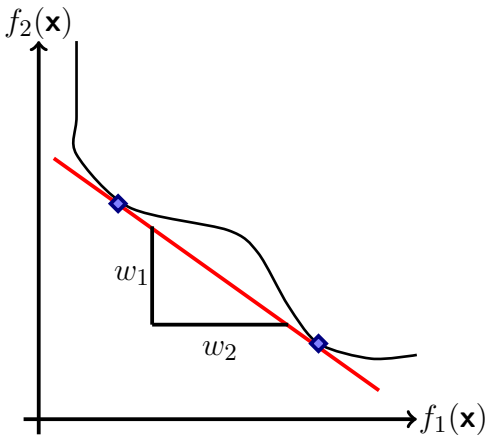


Abbildung 3.1: Gewichtete Summe bei nicht konvexer Pareto-Front

Abbildung 3.1 zeigt die Arbeitsweise der Gewichtete-Summen-Methode bei nicht konvexen Pareto-Fronten. Punkte auf dem konkaven Bereich der Pareto-Front sind für das Verfahren nicht erreichbar.

Im Folgenden soll nun ein gradientenbasiertes multikriterielles Optimierungsverfahren vorgestellt werden, dass in jedem Iterationsschritt eine Verbesserung bezüglich aller Ziele garantiert und auch Lösungen in konkaven Bereichen der Pareto-Front ermitteln kann.

3.1 MGDA-Verfahren

Gradientenbasierte multikriterielle Optimierung war bisher nur mit Hilfe von Ersatzzielfunktionen möglich. Der "Multiple Gradient Descent Algorithm" (MGDA) wurde 2009 von Desideri vorgestellt und wird in [Des09], [Des12] und [Des13] beschrieben. Der MGDA ermittelt aus den Gradienten der einzelnen Zielfunktionen im aktuellen Iterationspunkt eine gemeinsame Suchrichtung, die lokal alle Ziele verbessert. Der Prozess wird fortgesetzt bis keine weitere gemeinsame Abstiegsrichtung ermittelt werden kann.
Die grundlegende Idee des Verfahrens besteht darin, dass eine solche gemeinsame Abstiegsrichtung für alle Punkte existiert, die nicht paretooptimal sind. Dies folgt direkt aus der Definition der Pareto-Optimalität, da ein Punkt $\mathbf{x}$, für den eine gemeinsame Abstiegsrichtung $\mathbf{d}$ aller Ziele existiert, vom Punkt $\mathbf{x}+\epsilon\mathbf{d}$ für ein $\epsilon > 0$ dominiert wird. Umgekehrt ist ein Punkt, für den keine gemeinsame Abstiegsrichtung existiert, paretooptimal, da sich in diesem Fall in jeder Richtung der Zielfunktionswert bezüglich mindestens eines Zieles verschlechtert. Auch bei diesen Verfahren werden zunächst nur unrestringierte Probleme betrachtet.

3.1.1 Suchrichtung

Ziel des Verfahrens ist es, für den aktuellen Iterationspunkt einen Vektor **d** zu ermitteln, der bezüglich aller Zielfunktionen eine Abstiegsrichtung darstellt und somit der Bedingung

$$\nabla f_k(\mathbf{x})^T \mathbf{d} \leq 0, \quad k = 1, ..., K$$

genügt. Eine solcher Vektor existiert für alle Punkte **x**, die nicht paretooptimal sind. Dieser Fakt motiviert eine alternative Definition der Pareto-Optimalität (vgl.[Des09])

Definition 3.2 Ein Punkt $\mathbf{x}^*$ ist paretooptimal bezüglich der Zielfunktionen $f_k(\mathbf{x})$, $k = 1, ..., K$, wenn eine **positive** Linearkombination der Zielfunktionsgradienten existiert, sodass

$$\sum_{k=1}^{K} \alpha_k \nabla f_k(x^*) = 0, \quad \alpha_k \geq 0, \quad \sum_{k=1}^{K} \alpha_k = 1$$

Ein ausführlicher Nachweis der Äquivalenz von Definition 3.2 zur klassischen Definition der Pareto-Optimalität wird in [Des09] erbracht. Eine derartige Kombination von Vektoren wird als konvexe Linearkombination bezeichnet. Die Menge aller konvexen Linearkombinationen gegebener Vektoren bilden ihre konvexe Hülle.

Definition 3.3 Sei $\{\mathbf{x}_1, ..., \mathbf{x}_k\}$ eine Menge von Vektoren des $\mathbb{R}^n$. Die konvexe Hülle dieser Vektoren ist die Menge

$$X = \left\{ \sum_{i=1}^{k} \alpha_i \mathbf{x}_i | \mathbf{x} \in \mathbb{R}^n, \, \alpha_i \geq 0, \, \sum_{i=1}^{k} \alpha_i = 1 \right\} \tag{3.1}$$

aller konvexen Linearkombinationen.

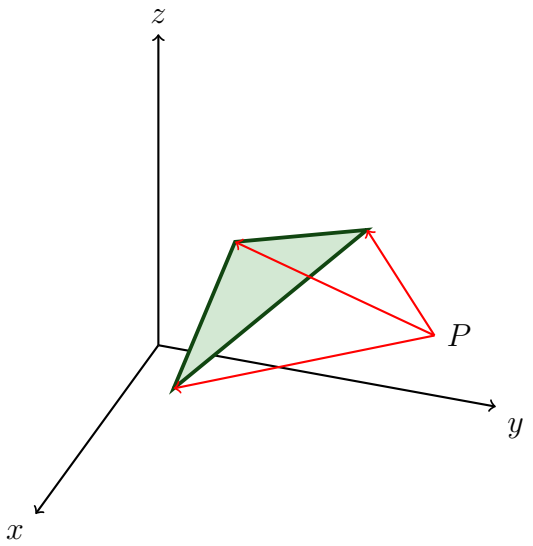


Abbildung 3.2: Konvexe Hülle der Endpunkte dreier Vektoren

Abbildung 3.2 zeigt drei vom Punkt P ausgehende Vektoren und ihre konvexe Hülle (grünes Dreieck). Die konvexe Hülle ist aufgrund ihrer Konstruktion abgeschlossen und konvex.

Bemerkung 3.4 Wie schon in Abbildung 3.2 zu sehen, wird die konvexe Hülle immer relativ zum aktuellen Iterationspunkt betrachtet, da auch die später verwendeten Gradienten relativ zum aktuellen Punkt sind.

Satz 3.5 *Innerhalb der konvexen Hülle existiert nach [Des09] stets ein eindeutig bestimmbares Element ω mit minimaler Norm und es gilt $\forall \boldsymbol{x} \in X : \langle \boldsymbol{x}, \omega \rangle \geq \|\omega\|^2$.*

Beweis: Seien $\mathbf{x}$, $\omega \in \widetilde{X}$ und ω das Element minimaler Norm. Dann ist aufgrund der Konvexität $\forall \epsilon \in [0,1]$ auch $\omega + \epsilon(\mathbf{x} - \omega) \in \widetilde{X}$. Aufgrund der minimalen Norm von ω gilt weiter

$$\begin{aligned} \|\omega + \epsilon(\mathbf{x} - \omega)\|^2 - \|\omega\|^2 = 2\epsilon\langle \mathbf{x} - \omega, \omega \rangle + \epsilon^2 \langle \mathbf{x} - \omega, \mathbf{x} - \omega \rangle \geq 0 \\ \langle \mathbf{x} - \omega, \omega \rangle \geq 0 \\ \langle \mathbf{x}, \omega \rangle \geq \langle \omega, \omega \rangle \end{aligned}$$

□

Dieses Konzept soll nun auf das multikriterielle Optimierungsproblem übertragen werden. Aus der Erkenntnis von Satz 3.5 folgt, dass für alle Vektoren innerhalb der konvexen Hülle das Skalarprodukt mit dem Element minimaler Norm größer Null ist. Somit gilt für das Skalarprodukt $\langle \mathbf{x}, (-\omega) \rangle$, das es stets kleiner Null ist, was der Bedingung für eine gemeinsame Abstiegsrichtung entspricht. Interpretiert man die Vektoren aus Definition 3.3 als Gradienten der einzelnen Zielfunktionen und das Element minimaler Norm als negative Suchrichtung $\mathbf{d} = -\omega$, ergibt sich

$$\forall \mathbf{x} \in \widetilde{X} : \langle \mathbf{x}, \mathbf{d} \rangle = -\langle \mathbf{x}, \omega \rangle \leq -\|\omega\|^2 \leq 0$$

Hierbei sind insbesondere die Gradienten, die X aufspannen, selbst Elemente von X und somit

$$\nabla f_k(\mathbf{x})^T \mathbf{d} = \langle \nabla f_k(\mathbf{x}), \mathbf{d} \rangle \leq -\|\omega\|^2 \leq 0, \quad k = 1, ..., K. \tag{3.2}$$

Aufgrund dieses Resultats ist $-\omega$ bzw. $\mathbf{d}$ Abstiegsrichtung bezüglich aller Zielfunktionen.

Bemerkung 3.6 Bei der Implementierung ist es wichtig, die Gradienten vorher geeignet zu skalieren, da sonst kurze Gradienten bei der Bestimmung von ω als Element minimaler Norm größeren Einfluss haben.

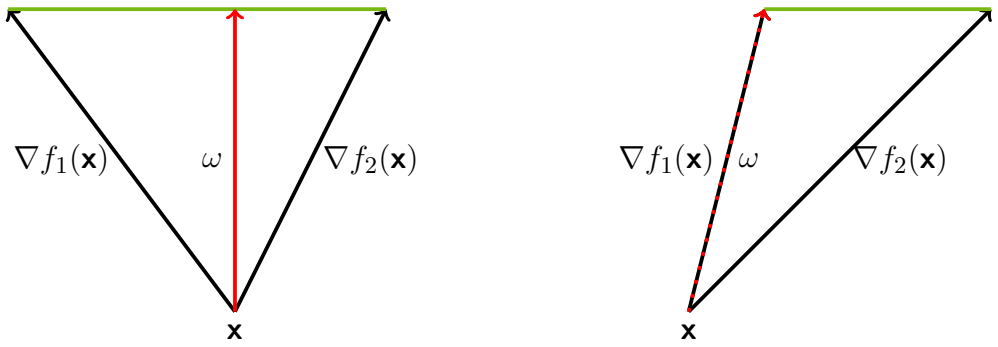


Abbildung 3.3: Element ω minimaler Norm

Abbildung 3.3 zeigt die Bestimmung von ω als Element minimaler Norm der konvexen Hülle (grüne Gerade). Kürzere Gradienten haben dabei einen stärkeren Einfluss auf die Suchrichtung. Das rechte Bild zeigt den Extremfall $\omega = \nabla f_1(\mathbf{x})$. Eine geeignete Skalierung unterbindet diesen Effekt.

Im Weiteren werden die skalierten Gradienten als $\mathbf{g}_k$ mit $\mathbf{g}_k = \dfrac{\nabla f_k(\mathbf{x})}{S_k}$ bezeichnet.

Eine Möglichkeit für die Wahl der Skalierungsfaktoren ist z.B. $S_k = \|\nabla f_k(\mathbf{x})\|$. Dies entspricht einer einfachen Normierung der Gradienten. Eine weitere Möglichkeit der Skalierung wird in [Des13] durch

$$S_k = \frac{\|\nabla f_k(\mathbf{x})\|^2}{\langle [\nabla^2 f_k(\mathbf{x})]^{-1} \nabla f_k(\mathbf{x}), \nabla f_k(\mathbf{x}) \rangle}$$

gegeben. Diese Wahl entspricht einer Projektion der Newton-Richtung in Richtung des Gradienten.

Mit Hilfe dieser Erkenntnisse lässt sich die Suche nach einer gemeinsamen Abstiegsrichtung auf die Berechnung des eindeutig bestimmbaren Elementes minimaler Norm innerhalb der konvexen Hülle der Gradienten zurückführen. Die Berechnung von ω als

$$\omega = \min_{\alpha_1,\ldots,\alpha_K} \left(\sum_{k=1}^{K} \alpha_k \mathbf{g}_k \right) \quad \alpha_k \geq 0 \quad \sum_{k=1}^{K} \alpha_k = 1 \tag{3.3}$$

stellt ein eigenes Optimierungsproblem dar, welches in jedem Iterationsschritt gelöst werden müsste. Der Aufwand zur Lösung von (3.3) erweist sich als zu groß, um ihn in jedem Schritt eines iterativen Verfahrens zu betreiben. Es existiert jedoch eine wesentlich einfachere Lösung von (3.3) für den Spezialfall, dass die Vektoren $\mathbf{g}_k$ zueinander orthogonal sind. In diesem Fall lässt sich durch einfaches Ausrechnen zeigen, dass gilt

$$\omega = \sum_{k=1}^{K} \alpha_k \mathbf{u}_k \overset{\mathbf{u}_j \perp \mathbf{u}_k \ \forall j \neq k}{\Longrightarrow} \|\omega\|^2 = \sum_{i=k}^{K} \alpha_k^2 \|\mathbf{u}_k\|^2. \tag{3.4}$$

Unter dieser Voraussetzung vereinfacht sich die analytische Berechnung der α_k über die Lagrange-Funktion. Aus der Optimierungsaufgabe

$$\begin{aligned} \|\omega\|^2 &\to min \\ \sum_{k=1}^{K} \alpha_k - 1 &= 0 \end{aligned}$$

ergibt sich nach Definition 1.12 die Lagrange-Funktion

$$\begin{aligned} L(\alpha, \lambda) &= \|\omega\|^2 - \lambda \left(\sum_{k=1}^{K} \alpha_k - 1 \right) \\ &= \sum_{k=1}^{K} \alpha_k^2 \|\mathbf{u}_k\|^2 - \lambda \left(\sum_{k=1}^{K} \alpha_k - 1 \right) \end{aligned}$$

Die rot gekennzeichnete Ersetzung wird durch Gleichung (3.4) ermöglicht. Diese Minimierung kann nun durch Bildung der partiellen Ableitungen und anschließendem Null setzen gelöst werden.

$$\begin{aligned} 0 &= \frac{\partial L}{\partial \alpha_k} = 2\|\mathbf{u}_k\|^2 \alpha_k - \lambda \Longrightarrow \alpha_k = \frac{\lambda}{2\|\mathbf{u}_k\|^2} \\ 0 &= \frac{\partial L}{\partial \lambda} = \left(\sum_{k=1}^{K} \alpha_k - 1 \right) \end{aligned}$$

Daraus erhält man durch Einsetzen

$$\begin{aligned} \frac{\lambda}{2} &= \frac{1}{\sum_{k=1}^{K} \frac{1}{\|\mathbf{u}_k\|^2}} \\ \alpha_k &= \frac{1}{\langle \mathbf{u}_k, \mathbf{u}_k \rangle \sum_{j=1}^{K} \frac{1}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle}} < 1, \quad k = 1, ..., K \end{aligned} \tag{3.5}$$

Da die Aufgabe allgemein gelöst wurde, ist Gleichung (3.5) für eine beliebige Anzahl von orthogonalen Vektoren einsetzbar.

Mit dieser Erkenntnis liegt es nahe, die Gradienten zu orthogonalisieren, um damit eine gemeinsame Abstiegsrichtung nach dem eben gezeigten Schema zu bestimmen. Eine Möglichkeit dafür bietet das Orthogonalisierungsverfahren nach Gram-Schmidt (vgl.[Mei08]).

Man berechnet

$$\mathbf{u}_1 = \mathbf{g}_1$$
$$\mathbf{u}_k = \frac{\mathbf{g}_k - \sum_{j<k} \frac{\langle \mathbf{u}_j, \mathbf{g}_k \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle} \mathbf{u}_j}{1 - \sum_{j<k} \frac{\langle \mathbf{u}_j, \mathbf{g}_k \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle}}, \quad k = 2, ..., K$$

Die spezielle Normierung sichert, dass die konvexe Hülle der orthogonalisierten Vektoren in der gleichen Hyperebene liegt, wie die der Gradienten.

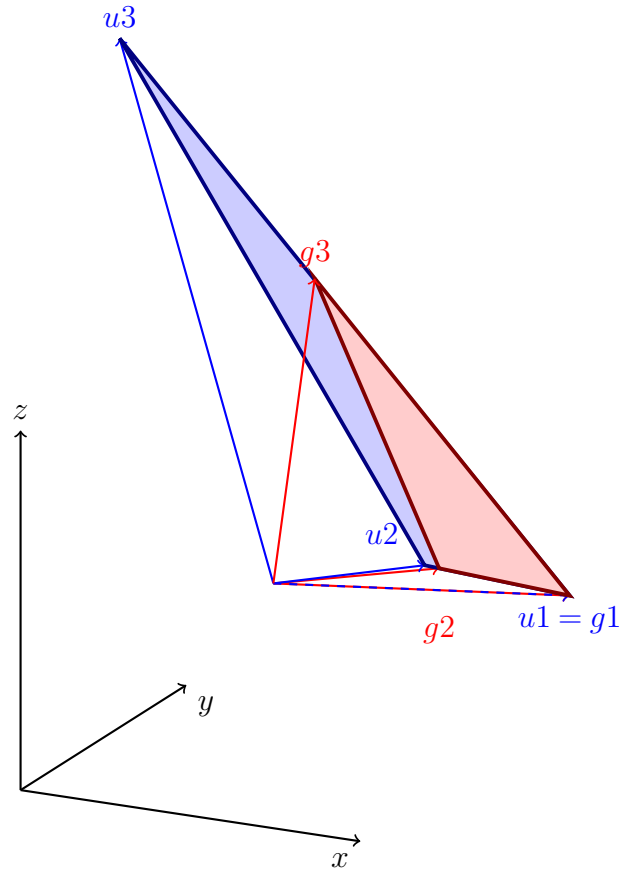


Abbildung 3.4: Orthogonalisierung der Gradienten

Abbildung 3.4 zeigt die konvexen Hüllen von Gradienten $\mathbf{g}_k$ und orthogonalisierten Vektoren $\mathbf{u}_k$. Durch die spezielle Normierung innerhalb des Gram-Schmidt Verfahrens liegen sie in der gleichen Hyperebene.

Mit Hilfe der orthogonalisierten Vektoren $\mathbf{u}_k$ werden nach (3.5) die Koeffizienten der Linearkombination bestimmt, um schließlich das Element minimaler Norm ω bzw. die

gemeinsame Abstiegsrichtung **d** zu bestimmen.

$$\omega = -\mathbf{d} = \sum_{i=1}^{p} \alpha_i \mathbf{u}_i$$

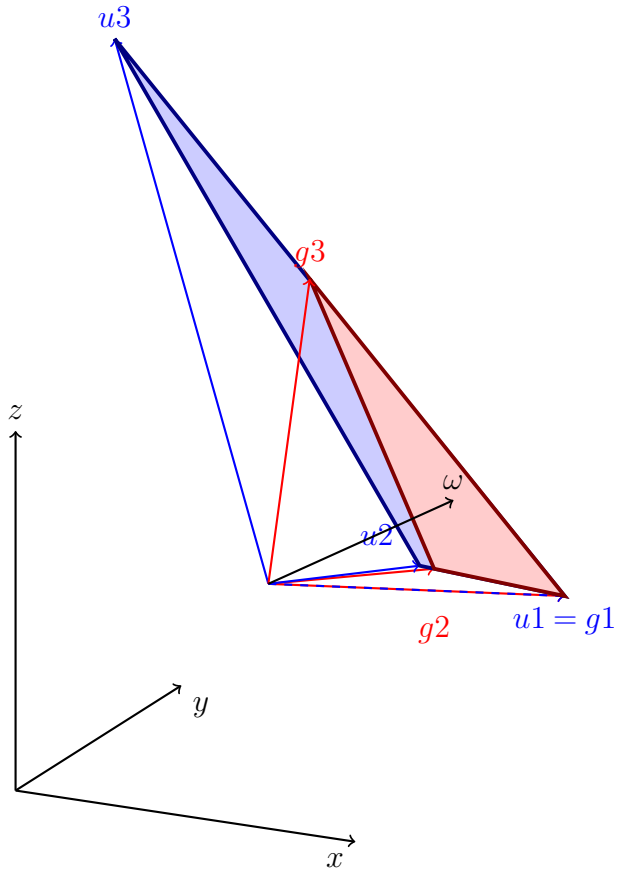


Abbildung 3.5: Element minimaler Norm

Der durch Linearkombination erzeugte Vektor ω bzw. $-\mathbf{d}$ steht senkrecht auf der konvexen Hülle der Gradienten. Bedingt durch die Konstruktion gilt zudem

$$\mathbf{g}_k^T \mathbf{d} = \mathbf{u}_k^T \mathbf{d} \leq 0, \quad \forall k = 1, ..., K.$$

Die durch die Orthogonalisierung vereinfachte Berechnung der Suchrichtung bringt jedoch einen Nachteil mit sich. Durch die Bestimmung von ω als positive Linearkombination der orthogonalisierten Vektoren $\mathbf{u}_k$ ist gesichert, dass ω innerhalb der konvexen Hülle der $\mathbf{u}_k$ liegt. Diese unterscheidet sich jedoch in einigen Fällen stark von der der $\mathbf{g}_k$, wie die folgende Abbildung zeigt.

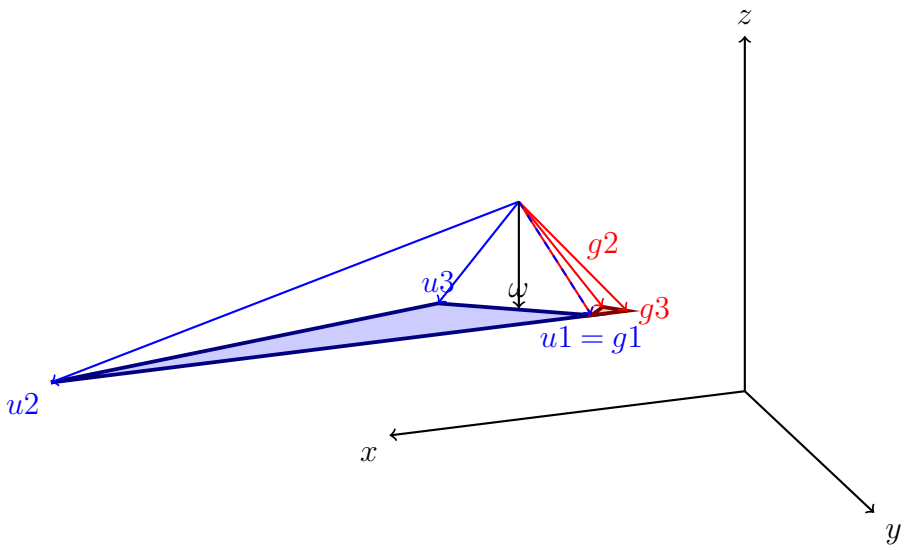


Abbildung 3.6: Ungünstige Konstellation der Gradienten

Auch in Abbildung 3.6 erfüllt $-\omega$ Bedingung (3.2) und ist somit lokal Abstiegsrichtung bezüglich aller Ziele. Durch die Position der Gradienten zueinander kommt es jedoch dazu, das ω außerhalb ihrer konvexen Hülle liegt. Eine solche Iteration endet in einem Zustand, in dem die Gradienten zwar linear abhängig sind, es jedoch keine **positive** Linearkombination der Form

$$\sum_{k=1}^{K} \alpha_k \mathbf{g}_k = 0, \quad \sum_{k=1}^{K} \alpha_k = 1$$

gibt, sodass die Bedingung der Pareto-Optimalität nicht erfüllt ist.

Der in Abbildung 3.6 beschriebene Fall tritt ein, wenn sich die Richtungen der einzelnen Gradienten nur wenig voneinander unterscheiden. Gerade in den ersten Schritten der Iteration, wenn der aktuelle Iterationspunkt noch weit von der Pareto-Front entfernt ist, beobachtet man dies häufig. Um dem vorzubeugen, besteht die Möglichkeit, die Reihenfolge der Orthogonalisierung vorteilhaft zu wählen und den Gram-Schmidt-Prozess bei hinreichend guter Approximation einer gemeinsamen Abstiegsrichtung vorzeitig abzubrechen.

3.1.2 Unvollständige Orthogonalisierung

Vor allem in den ersten Iterationsschritten ist es nicht sinnvoll, zu viel Berechnungszeit in die Bestimmung der Suchrichtung zu investieren, da der aktuelle Iterationspunkt im Allgemeinen noch weit von der Pareto-Front entfernt ist und deshalb häufig der am Ende von Abschnitt 3.1.1 beschriebene Fall eintritt. In [Des13] wird eine Variante beschrieben, die den Gram-Schmidt-Prozess vorzeitig abbricht, sobald die aktuelle Approximation von ω die Bedingung für eine Abstiegsrichtung bezüglich aller Ziele erfüllt. An dieser Stelle ist es sinnvoll, eine Hilfsmatrix $C \in \mathbb{R}^K \otimes \mathbb{R}^K$ mit

$$C_{kj} = \frac{\langle g_k, u_j \rangle}{\langle u_j, u_j \rangle}, \quad k = 2, ..., K, \quad j = 1, ..., k-1$$
$$C_{kK} = \sum_{j=1}^{K-1} C_{kj}, \quad k = 2, ..., K$$

sowie eine Abbruchkonstante $a \in [0, 1)$ einzuführen. Der Eintrag C_{kK} dient als Indikator für die Güte der aktuellen Approximation bezüglich des k-ten Gradienten. Die Konstante a legt fest, wie gut die Approximation sein muss, damit ein Abbruch des Gram-Schmidt-Prozesses erfolgt.

Die Matrix C wird im laufenden Gram-Schmidt-Prozess nach jedem neu berechneten orthogonalen Vektor aktualisiert. Der Wert s steht im Folgenden für die Anzahl der bereits orthogonalisierten Vektoren. Über die Bedingung

$$C_{kK} > a, \quad \forall k = s+1, ..., K \tag{3.6}$$

wird geprüft, ob die Approximation von ω aus den s bereits orthogonalisierten Vektoren gleichzeitig eine hinreichend gute Abstiegsrichtung für die $K - s$ noch nicht orthogonalisierten Gradienten ist. Ist dies nicht der Fall, wird ein weiterer orthogonaler Vektor aus dem Gradienten

$$g_m \ mit \ m = \underset{k=s+1,...,K}{argmin} (C_{kK})$$

bestimmt. Bei dieser Wahl entspricht g_m dem Gradienten, der bisher am schlechtesten approximiert wurde. Diese Strategie nutzt die vorhandenen Gradienten optimal aus und minimiert die Anzahl der notwendigen Orthogonalisierungen.

In der praktischen Anwendung zeigt sich, dass vor allem in den ersten Iterationen der Abbruch oft schon nach der Berechnung eines orthogonalen Vektors erfolgt. Nähert sich der Iterationspunkt schließlich der Pareto-Front, werden meist alle Gradienten zur Bestimmung von ω verwendet.

1: Setze $\mathbf{u}_1 = \mathbf{g}_1$ und $a \in [0,1)$
2: **for** $s = 1$ **to** $K-1$ **do**
3: Berechne $C_{ks} = \dfrac{\langle \mathbf{g}_k, \mathbf{u}_s \rangle}{\langle \mathbf{u}_s, \mathbf{u}_s \rangle} \quad k = s+1, ..., K$
4: Berechne $C_{kK} = \sum_{i=1}^{K-1} C_{ki} \quad k = 2, ..., K$
5: **if** $C_{kK} > a \quad \forall k = s+1, ..., K$ **then**
6: **return** $\{\mathbf{u}_1, ..., \mathbf{u}_s\}$
7: **end if**
8: Bestimme $m = \underset{i=s+1,...,K}{argmin} \, (C_{iK})$
9: Vertausche Zeilen m und $s+1$ der Matrix C sowie $\mathbf{g}_m$ und $\mathbf{g}_{s+1}$
10: Berechne $\mathbf{u}_{s+1} = \dfrac{\mathbf{g}_{s+1} - \sum_{i=1}^{s} C_{s+1,i} \mathbf{u}_i}{1 - C_{s+1,K}}$
11: **end for**
12: **return** $\{\mathbf{u}_1, ..., \mathbf{u}_K\}$

Algorithmus 4: unvollständiger Gram-Schmidt-Prozess (vgl.[Des13])

Ein vorzeitiger Abbruch durch Erfüllung der Bedingung in Zeile fünf von Algorithmus 4 verringert die Berechnungszeit, verletzt jedoch nicht die Bedingung an die gemeinsame Abstiegsrichtung. Für bereits orthogonalisierte Gradienten gilt

$$\langle \mathbf{g}_i, \omega \rangle = \langle \mathbf{u}_i, \omega \rangle \geq 0, \quad i = 1, ..., u$$

, d.h. ω erfüllt die Bedingung an eine Abstiegsrichtung. Die übrigen Gradienten lassen sich als Linearkombination der berechneten $\mathbf{u}_i$ plus einer Komponente $\mathbf{v}$, welche orthogonal zu allen $\mathbf{u}_i$ ist, darstellen.

$$\begin{aligned} \omega &= \sum_{i=1}^{s} \alpha_i \mathbf{u}_i \\ \mathbf{g}_k &= \sum_{i=1}^{s} C_{ki} \mathbf{u}_i + \mathbf{v}_k, \quad \mathbf{v}_k \perp \{\mathbf{u}_1, ..., \mathbf{u}_s\}, \quad k = s+1, ..., K \end{aligned}$$

Für die Bedingung der Abstiegsrichtung gilt damit

$$\begin{aligned} \langle \mathbf{g}_k, \omega \rangle &= \langle \sum_{i=1}^{s} C_{ki} \mathbf{u}_i + \mathbf{v}_k, \omega \rangle \\ &= \sum_{i=1}^{s} C_{ki} \langle \mathbf{u}_i, \omega \rangle + \underbrace{\langle \mathbf{v}, \omega \rangle}_{=0} \\ &\geq \sum_{i=1}^{s} C_{ki} \|\omega\|^2 = C_{kK} \|\omega\|^2 > a \|\omega\|^2 > 0, \quad k = s+1, ..., K \end{aligned}$$

Die positive Abbruchkonstante a sichert an dieser Stelle die Einhaltung der Bedingung an eine Abstiegsrichtung.

In [Des13] wird eine Verbesserung von Algorithmus 4 vorgeschlagen, bei der der erste Gradient $\mathbf{g}_1$ gezielt gewählt wird. Dazu setzt man

$$\mathbf{u}_1 = \mathbf{g}_k \;\; mit \;\; k = \underset{i=1,\ldots,K}{argmax} \left(\underset{j=1,\ldots,K}{min} \left(\frac{\langle \mathbf{g}_j, \mathbf{g}_i \rangle}{\langle \mathbf{g}_i, \mathbf{g}_i \rangle} \right) \right). \tag{3.7}$$

Diese Wahl entspricht dem Gradienten, der den allgemeinen "Trend" aller Gradientenrichtungen am besten widerspiegelt.

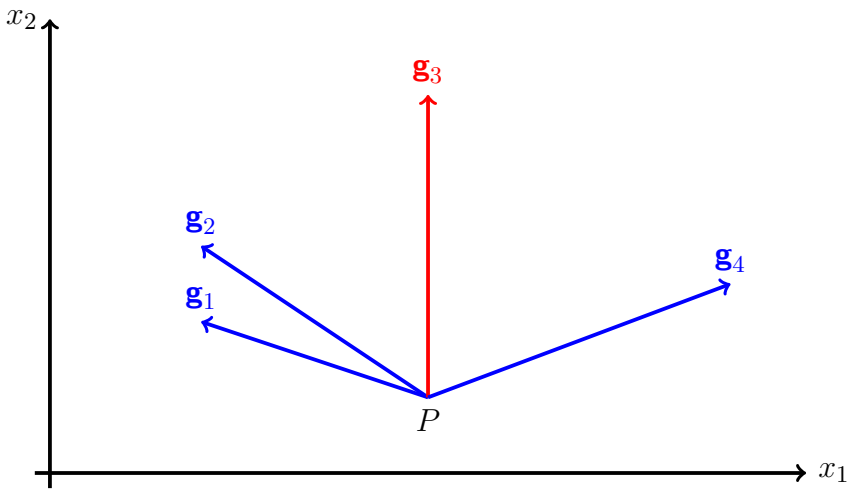


Abbildung 3.7: Wahl des ersten Gradienten

Abbildung 3.7 zeigt mögliche vier Gradienten am Punkt P. Bei Anwendung von Gleichung (3.7) wird in diesem Beispiel $\mathbf{g}_3$ ausgewählt, da für $\mathbf{g}_3$ selbst das kleinste Skalarprodukt mit einem anderen Gradienten immer noch positiv ist. Für alle anderen $\mathbf{g}_k$ lässt sich mindestens ein Gradient finden, mit dem das Skalarprodukt negativ ist.

Bemerkung 3.7 In [Zem08] wird darauf hingewiesen, dass das Gram-Schmidt-Verfahren für praktische Anwendungen nur bedingt einsetzbar ist, da es bei nahezu linear abhängigen Gradienten numerisch instabil werden kann. Ersatzweise sollte man daher auf eine QR-Zerlegung nach Givens oder Householder zurückgreifen.

3.1.3 Skalierung und Schrittweite

Nach der Bestimmung der Suchrichtung ist auch beim MGDA eine Schrittweitensteuerung erforderlich. Hierzu kann z.B. auf Liniensuchmethoden, wie etwa in 2.2.2 vorgestellt, zurückgegriffen werden. Nachteilig hierbei erweist sich, dass die Liniensuche in jedem Schritt bezüglich aller Zielfunktionen durchgeführt werden muss. Ein solches Vorgehen benötigt viel Rechenzeit und verlangsamt den Algorithmus. In [Des09] wird daher eine

Schrittweitensteuerung mit Hilfe von BFGS-Matrizen vorgeschlagen. Dazu wird während der gesamten Iteration für jede Zielfunktion eine BFGS-Matrix (wie in Abschnitt 2.2.1 beschrieben) mitgeführt und in jedem Schritt mit Gleichung (2.6) aufaddiert. Mit Hilfe dieser Matrix wird jede Zielfunktion entlang der gemeinsamen Suchrichtung durch ein quadratisches Modell der Form

$$f_k(\mathbf{x}+\rho_k\mathbf{d}) \approx f_k(\mathbf{x}) - \rho_k \nabla f_k(\mathbf{x})^T\mathbf{d} + \frac{\rho_k^2}{2}\mathbf{d}^T H_k \mathbf{d}$$

approximiert. Anschließend wird anhand dieses Modells die Verbesserung $\triangle f_k(\mathbf{x})$ des Zielfunktionswertes bestimmt und über eine Minimierung bezüglich ρ_k die optimale Schrittweite ermittelt.

$$\begin{aligned}
\triangle f_k(\mathbf{x}) = f_k(\mathbf{x}) - f_k(\mathbf{x}+\rho_k\mathbf{d}) &= \rho_k \nabla f_k(\mathbf{x})^T\mathbf{d} - \frac{\rho_k^2}{2}\mathbf{d}^T H_k\mathbf{d}, \quad k=1,...,K \\
\triangle f_k(\mathbf{x})\rho_k &= \nabla f_k(\mathbf{x})^T\mathbf{d} - \rho_k \mathbf{d}^T H_k \mathbf{d} \\
0 &= \nabla f_k(\mathbf{x})^T\mathbf{d} - \rho_k \mathbf{d}^T H_k \mathbf{d} \\
\rho_k &= \frac{\nabla f_k(\mathbf{x})^T\mathbf{d}}{\mathbf{d}^T H_k \mathbf{d}} = \frac{S_k \mathbf{g}_k^T \mathbf{d}}{\mathbf{d}^T H_k \mathbf{d}} = \frac{S_k \mathbf{d}^T \mathbf{d}}{\mathbf{d}^T H_k \mathbf{d}}
\end{aligned}$$

Als Schrittweite wird schließlich das Minimum

$$\rho^* = \min_k(\rho_k) = \min_k\left(\frac{S_k \mathbf{d}^T\mathbf{d}}{\mathbf{d}^T H_k \mathbf{d}}\right)$$

gewählt.

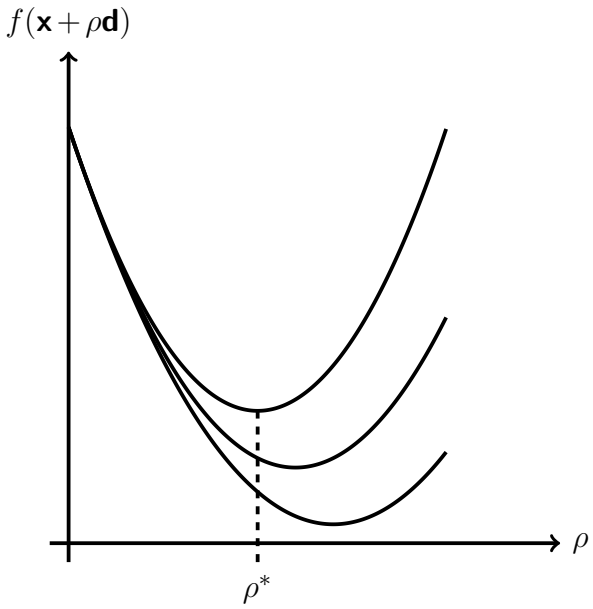


Abbildung 3.8: Maximierung der minimalen Verbesserung

Die Abbildung 3.8 zeigt eine mögliche quadratische Approximation dreier Zielfunktionen. Die Schrittweite wird so bestimmt, dass die Approximation mit der geringsten absoluten Verbesserung maximal verbessert wird.

Zusammen ergeben die Komponenten aus 3.1.2 und 3.1.3 eine Variante des MGDA-Verfahrens. An dieser Stelle sei erwähnt, das in jedem Iterationsschritt nur einmal die Zielfunktionsgradienten berechnet werden und sonst keine weiteren Funktionsauswertungen notwendig sind.

1: wähle Startpunkt $\mathbf{x}_0 \in \mathbb{R}^n$, $H_k = I$, $\forall k = 1, ..., K \quad t = 0 \quad a \in [0, 1)$
2: **repeat**
3: Bestimme alle Gradienten $\nabla f_k(\mathbf{x}_t) \quad k = 1, ..., K$
4: Berechne Skalierungen S_k und $\mathbf{g}_k = \dfrac{\nabla f_k(\mathbf{x}_t)}{S_k} \quad k = 1, ..., K$
5: Bestimme $l = \underset{i}{argmax} \left(\underset{j}{min} \left(\dfrac{\langle \mathbf{g}_j, \mathbf{g}_i \rangle}{\langle \mathbf{g}_i, \mathbf{g}_i \rangle} \right) \right) \quad i, j = 1, ..., K$
6: Setze $\mathbf{u}_1 = \mathbf{g}_l$
7: Berechne orthogonalisierte Vektoren nach Algorithmus 4
8: Berechne Koeffizienten α_k
9: Berechne Suchrichtung $\mathbf{d}_t$
10: Berechne Schrittweite $\rho_t^* = \underset{k}{min} \left(\dfrac{S_k \mathbf{d}_t^T \mathbf{d}_t}{\mathbf{d}_t^T H_k \mathbf{d}_t} \right)$
11: Berechne Schritt $\mathbf{x}_{t+1} = \mathbf{x}_t + \rho_t^* \mathbf{d}_t$
12: Berechne BFGS Update der Hesse Matrizen $(H_k)_{t+1} \quad k = 1, ..., K$
13: Setze $t = t + 1$
14: **until** $\|\mathbf{d}_{t-1}\| < \epsilon$
15: **return** $\mathbf{x}_t$

Algorithmus 5: MGDA-Verfahren

Konvergenzeigenschaften

Gradientenbasierte Optimierungsverfahren haben die Eigenschaft, sowohl gegen globale, als auch gegen lokale Minima zu konvergieren. Im Bereich der multikriteriellen Optimierung existieren, ebenso wie bei einkriteriellen Problemen, lokale Optima in Form von lokalen Pareto-Fronten. Diese bestehen aus Punkten, für die es lokal keine gemeinsame Verbesserungsrichtung aller Zielfunktionen gibt, die aber global von anderen Punkten dominiert werden.

Beispiel 3.8

$$f_1(x,y) = \frac{x^4}{8} - x^2 + 2 + y^2 \to min, \quad f_2(x,y) = (x-1)^2 + (y-3)^2 \to min$$

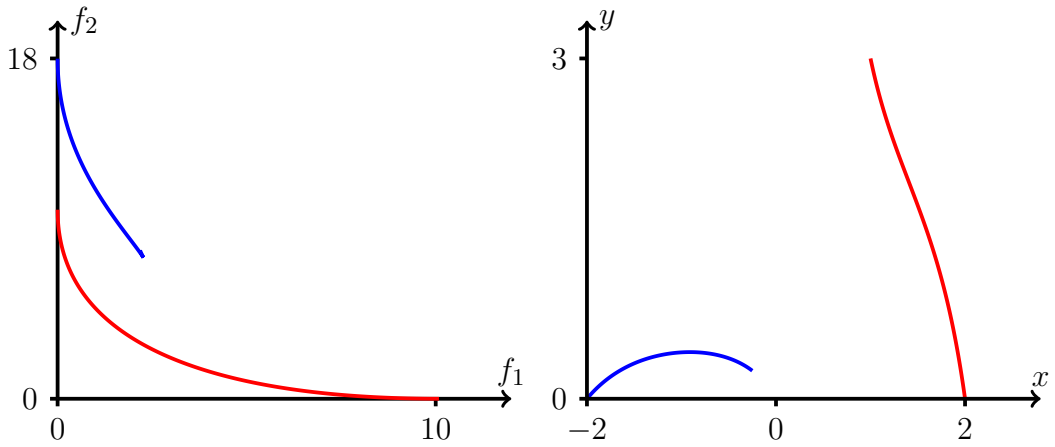


Abbildung 3.9: Lokale und globale Pareto-Front

In diesem Beispiel tritt eine lokale Pareto-Front (blaue Kurve) auf. Für die Punkte auf der blauen Kurve gibt es keine lokale Verbesserungsrichtung. Im linken Bild erkennt man jedoch, dass die blaue Kurve von der roten dominiert wird. Der MGDA-Algorithmus konvergiert in diesem Beispiel für Startpunkte nahe der blauen Kurve gegen diese. Die Punkte der lokalen Pareto-Front erfüllen das Abbruchkriterium und der Algorithmus stoppt.

Restringierte Optimierung

Der Umgang mit Gleichungsnebenbedingungen ist wie beim BFGS-Verfahren mittels einer Straffunktion wie der Augmented-Lagrange-Funktion möglich. Bei multikriteriellen Problemen wird der Strafterm auf jede Zielfunktion aufaddiert. Bei der praktischen Implementierung erweist sich die dynamische Anpassung des Strafparameters als schwierig, da ein zu langsames Wachstum zu vorzeitiger Konvergenz und ein zu schnelles Wachstum zu sehr kleinen Schrittweiten führen kann. Der Umgang mit Ungleichungsrestriktionen erweist sich als sehr aufwändig. Zu diesem Problem sind gesonderte Betrachtungen notwendig, die über den Umfang dieser Arbeit hinausgehen.

3.2 Einkriterielle Optimierung mit dem MGDA

Obwohl die primären Anwendungsgebiete des MGDA in der multikriteriellen Optimierung liegen, ist er ebenfalls bei einkriteriellen Problemen einsetzbar. Dabei entspricht der MGDA einem einfachen Gradientenverfahren, das die negative Gradientenrichtung als Suchrichtung verwendet. Vergleiche bezüglich der Laufzeit zeigen jedoch, dass die in dieser Arbeit vorgestellte Schrittweitensteuerung der Liniensuche des BFGS-Verfahrens im Allgemeinen unterlegen ist. Das liegt darin begründet, dass die Liniensuche in jedem Schritt einen viel größeren Bereich abdecken kann, als das lokal begrenzte quadratische Ersatzmodell in der MGDA Schrittweite.

4 Numerische Tests

In diesem Abschnitt sollen die vorgestellten Verfahren anhand einiger analytischer Testfunktionen verglichen werden. Im Bereich der einkriteriellen Optimierung werden BFGS-Verfahren und MGDA mit dem SQP-Verfahren (Sequentiell Quadratic Programming) verglichen. Bei multikriteriellen Aufgaben tritt der MGDA gegen die evolutionären Algorithmen IBEA (Indicator Based Evolutionary Algorithm) und SPEA (Strength Pareto Evolutionary Algorithm) an.

Die Bewertung der Verfahren bezüglich der einkriteriellen Probleme erfolgt für jeden der gewählten Startpunkte anhand des bekannten optimalen Zielfunktionswertes. Ein Vergleich der Ergebnisse der multikriteriellen Aufgaben gestaltet sich hingegen schwieriger. Da sich die Arbeitsweise des MGDA stark von der der evolutionären Verfahren unterscheidet, ist es nicht möglich, klassische Vergleichskriterien wie z.B. *durchschnittliche relative Dominanz* zu verwenden. Als Kriterium für die Qualität der gefundenen Lösungen soll hier die Vereinigung der Pareto-Mengen der einzelnen Verfahren herangezogen werden. Dazu werden die Pareto-Mengen der einzelnen Verfahren zu einer Menge vereinigt, von der anschließend wieder die Pareto-Menge gebildet wird. Daraus kann der Anteil der einzelnen Verfahren an der Mengenvereinigung nach folgendem Algorithmus bestimmt werden.

1: Setze $\{P_1, ..., P_k\}$ als Pareto-Mengen der einzelnen Verfahren sowie $q_i = 0 \;\; i = 1, ..., k$.
2: Berechne die Vereinigung $\bigcup_{i=1}^{k} P_i$ sowie deren nichtdominierte Menge P^*.
3: **for** $i = 1$ **to** k **do**
4: **for** $j = 1$ **to** $|P_i|$ **do**
5: **if** $P_i(j) \in P^*$ **then**
6: $q_i = q_i + 1$
7: **end if**
8: **end for**
9: **end for**
10: **return** Anteile $\dfrac{q_i}{|P_i|} \;\; i = 1, ..., k$

Algorithmus 6: Anteil an der Vereinigten Pareto-Menge

Dieses Kriterium bewertet die Güte der einzelnen Lösungsmengen bezüglich der Lösungen, die mit den anderen Verfahren erreicht wurden. Die folgende Abbildung soll dieses Konzept veranschaulichen.

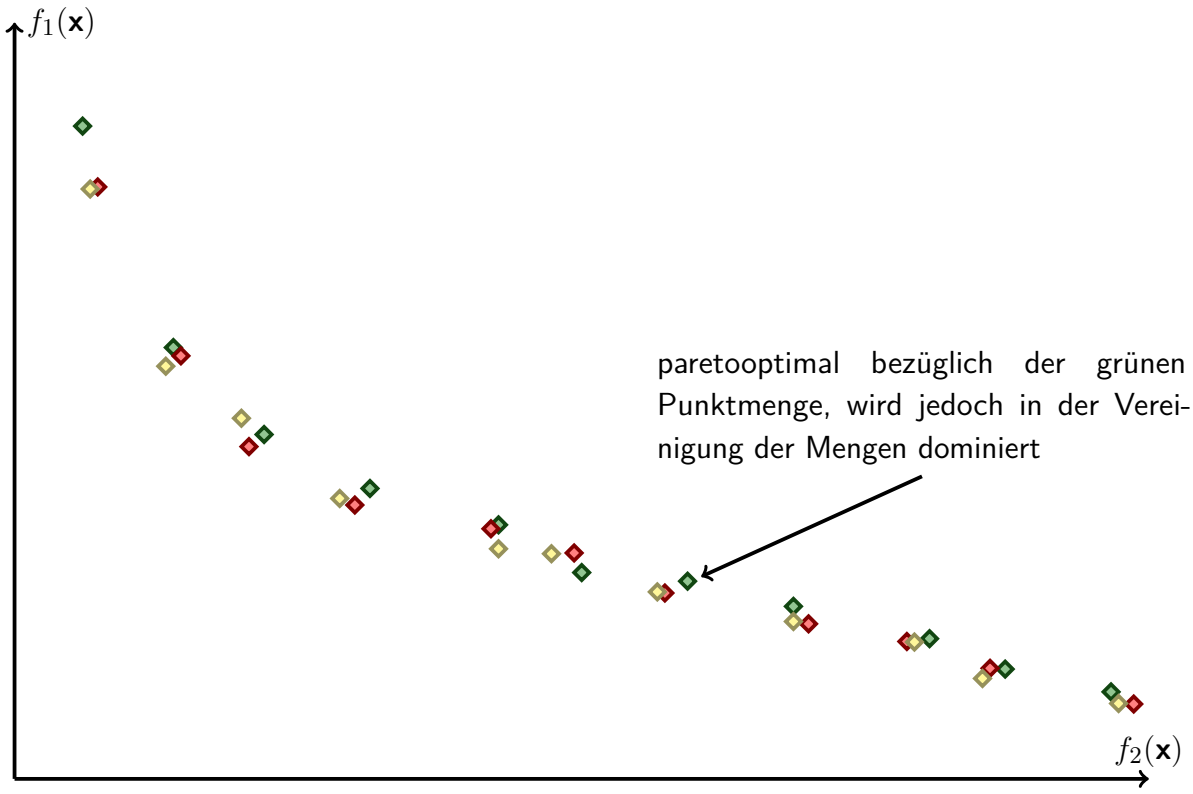


Abbildung 4.1: Vereinigung dreier Pareto-Fronten

In Abbildung 4.1 stellen drei Pareto-Mengen die Lösung einer Optimierungsaufgabe mit drei verschiedenen Verfahren dar. Bildet man die nichtdominierte Menge der Vereinigung, fallen einige der vorher paretooptimalen Punkte weg. In diesem konkreten Beispiel erhält die grüne Punktmenge die schlechteste Bewertung, da die meisten ihrer Punkte in der Vereinigung von anderen dominiert werden.

Für die folgenden Tests wurden bei den evolutionären Verfahren 65 Generationen mit je 200 Individuen berechnet. Der MGDA arbeitet 128 Punkte auf einem regelmäßigen Gitter über dem Parameterbereich ab. Bei allen anderen Verfahren wurden die Standardparameter beibehalten.

Bemerkung 4.1 Da die evolutionären Algorithmen SPEA und IBEA nicht deterministisch arbeiten und somit ein einzelner Durchlauf der Optimierung nicht aussagekräftig ist, wurden mit diesen Verfahren jeweils 30 Optimierungen pro Testfunktion durchgeführt und Mittelwerte über die Resultate gebildet.

4.1 Einkriterielle Testfunktionen

Himmelblau Funktion

Eines der bekanntesten Testprobleme für Optimierungsalgorithmen ist die Himmelblau Funktion.

$$f(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$
$$x_1, x_2 \in [-8,8]$$

Diese Funktion besitzt vier globale Minima bei $(3,2)^T$, $(-2.805,3.131)^T$, $(-3.779,-3.283)^T$, $(3.584,-1.848)^T$ sowie ein lokales Maximum bei $(-0.271,-0.923)^T$.

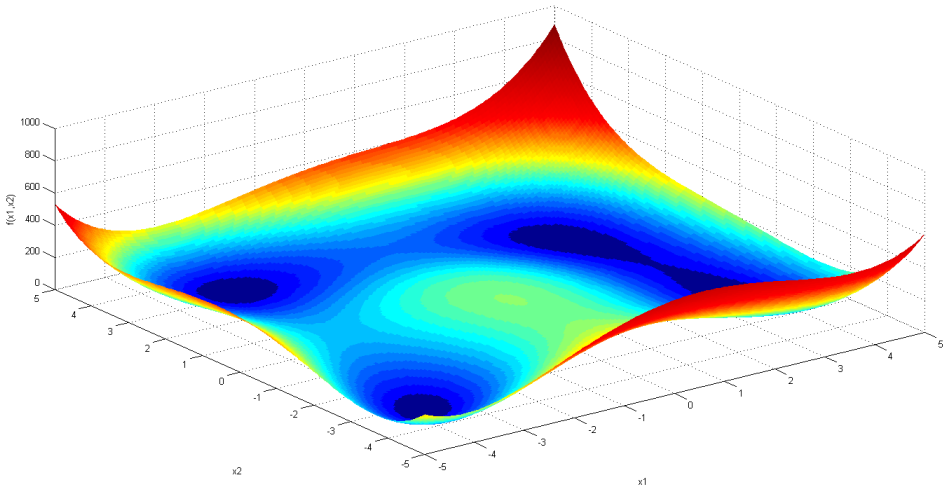


Abbildung 4.2: Himmelblau-Funktion

Startpunkt	BFGS	MGDA	SQP
$(0,0)$	$(3,2)$	$(3.584,-1.848)$	$(3,2)$
$(-4,3)$	$(-2.805,3.131)$	$(3,2)$	$(3.584,-1.848)$
$(3,3)$	$(3,2)$	$(-3.779,-3.283)$	$(2.999,1.999)$
$(-6,-1)$	$(3.584,-1.848)$	$(3,2)$	$(-3.796,-3.275)$
$(7,-7)$	$(3.584,-1.848)$	$(-2.805,3.131)$	$(3.001,2.001)$
$(5,0)$	$(-3.779,-3.283)$	$(3,2)$	$(3.582,-1.837)$

Bei diesem Beispiel streben alle Verfahren stets eines der globalen Minima an. Dies war zu erwarten, da die Funktion keine lokalen Minima besitzt, in welchen die Verfahren konvergieren können. Durch die verschiedenen Suchstrategien der einzelnen Methoden kommt es zu Unterschieden im Verlauf der Iterationen. Bei keinem der Verfahren lässt sich vom gewählten Startpunkt auf das erreichte Minimum schließen.

Welliges Tal

Diese Funktion besitzt ein globales Minimum bei $(23.026, 0)$. Die Schwierigkeit bei dieser Aufgabe besteht darin, die kleinen Wellen auf dem Weg zum Minimum zu überwinden.

$$f(x_1, x_2) = e^{\frac{x_1}{10}} - x_1 + \frac{x_2^2}{10} + 5sin^2(x_2)$$
$$x_1 \in [-25, 35]$$
$$x_2 \in [-30, 30]$$

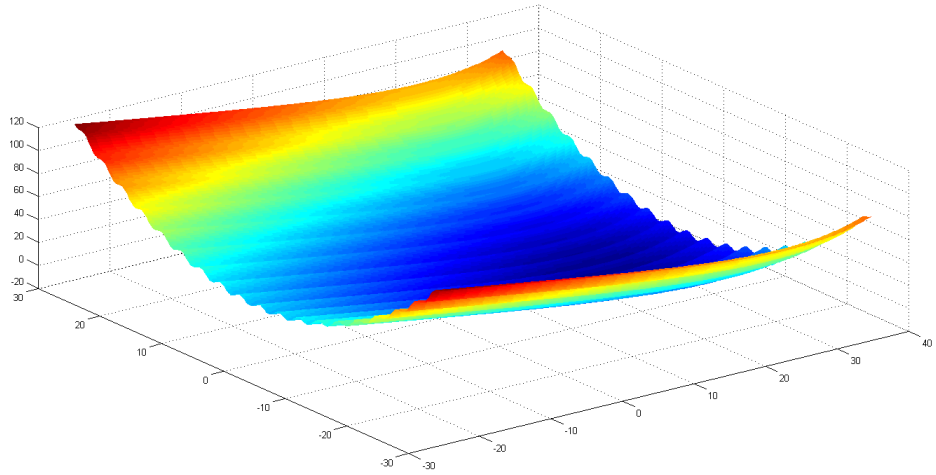


Abbildung 4.3: Welliges Tal

Startpunkt	BFGS	MGDA	SQP
$(0, 0)$	$(23.026, 0)$	$(23.026, 0)$	$(23.025, 0)$
$(-20, 25)$	$(23.026, 0)$	$(23.026, 0)$	$(23.026, 3.08)$
$(-25, 0)$	$(23.026, 0)$	$(23.026, 6.16)$	$(23.026, 0)$
$(30, 30)$	$(23.026, 0)$	$(23.026, -6.16)$	$(23.026, 24.452)$
$(0, 25)$	$(23.026, 0)$	$(23.026, 21.475)$	$(23.025, 15.377)$

Bei diesem Test schneidet der BFGS-Algorithmus am besten ab. Die Liniensuche zur Schrittweitenbestimmung scheint hier sehr gut zu funktionieren. Die beiden anderen Verfahren haben Schwierigkeiten beim Überwinden der kleinen Wellen und konvergieren oft in lokalen Minima.

Beale Funktion

Diese Funktion besitzt ein globales Minimum bei $(3, 0.5)$. Bei diesem Problem besteht die Möglichkeit, dass der Algorithmus im lokalen Tal im hinteren Teil der Abbildung 4.4 konvergiert.

$$f(x_1, x_2) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$
$$x_1, x_2 \in [-4.5, 4.5]$$

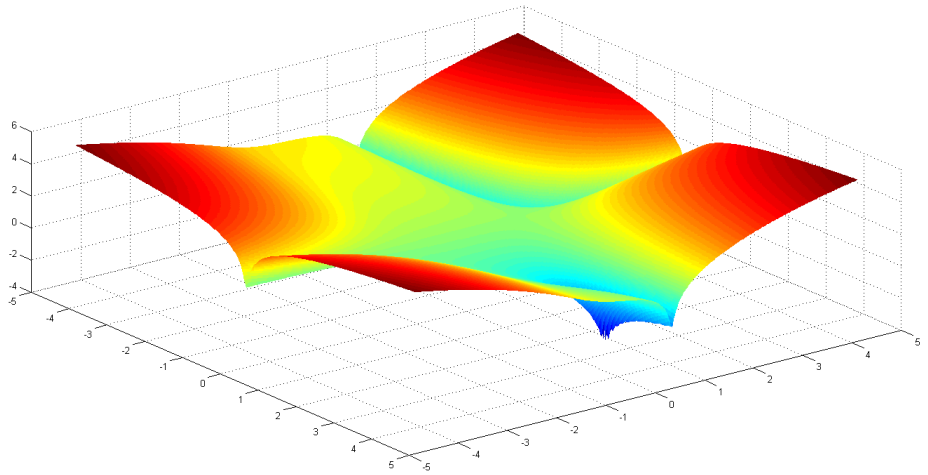


Abbildung 4.4: Beale-Funktion

Startpunkt	BFGS	MGDA	SQP
$(0, 0)$	$(3, 0.5)$	$(2.999, 0.499)$	$(3.001, 0.5)$
$(-4, -4)$	$(-4.5, 1.186)$	$(2.999, 0.499)$	$(3, 0.5)$
$(-3, 4)$	$(-4.5, 1.186)$	$(2.999, 0.499)$	$(2.999, 0.499)$
$(1, -3)$	$(-4.5, 1.186)$	$(0.019, -4.5)$	$(2.999, 0.499)$
$(3.5, 3)$	$(0.023, -4.5)$	$(-4.5, 1.186)$	$(3, 0.5)$

Bei diesem Test hat der BFGS-Algorithmus große Probleme beim Auffinden des globalen Minimums. Er erreicht die gesuchte Lösung nur von einem der gewählten Startpunkte aus. Auch der MGDA hat Schwierigkeiten, das im Hintergrund von Abbildung 4.4 erkennbare Tal zu überwinden. SQP erzielt hier die besten Ergebnisse.

Goldstein-Price Funktion

Diese Funktion besitzt ein globales Minimum bei $(0,-1)$. Innerhalb des blauen Tals befinden sich zwei lokale Minima. Läuft die Iteration zu schnell in das Tal, konvergieren die Verfahren in den lokalen Minima.

$$\begin{aligned} f(x_1,x_2) =& (1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)) \\ & (30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)) \\ x_1,x_2 \in & [-2,2] \end{aligned}$$

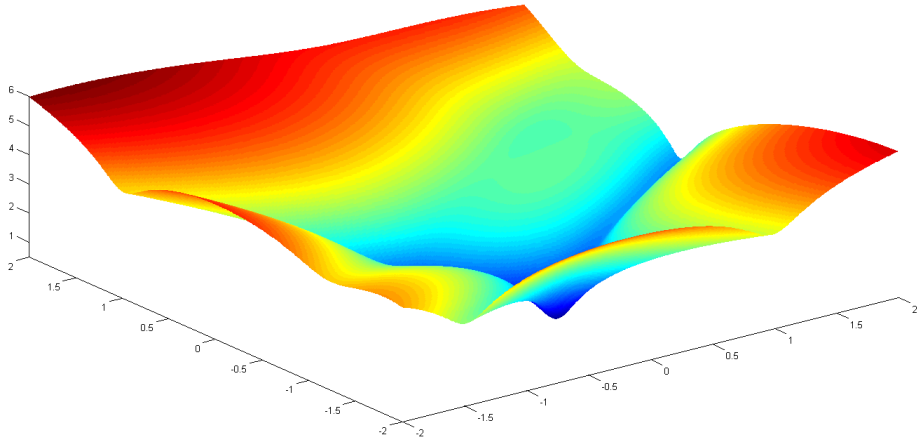


Abbildung 4.5: Goldstein-Price Funktion

Startpunkt	BFGS	MGDA	SQP
$(0,0)$	$(0,-0.998)$	$(-0.548,-0.401)$	$(0,-1)$
$(-2,-2)$	$(-0.002,-1)$	$(0,-1)$	$(0,-1.001)$
$(-2,2)$	$(-0.001,-0.999)$	$(0,-1.001)$	$(-0.553,-0.398)$
$(2,-2)$	$(0.002,-0.999)$	$(0,-1)$	$(0,-1.001)$
$(2,2)$	$(1.998,0.297)$	$(0,-1.002)$	$(1.989,0.304)$

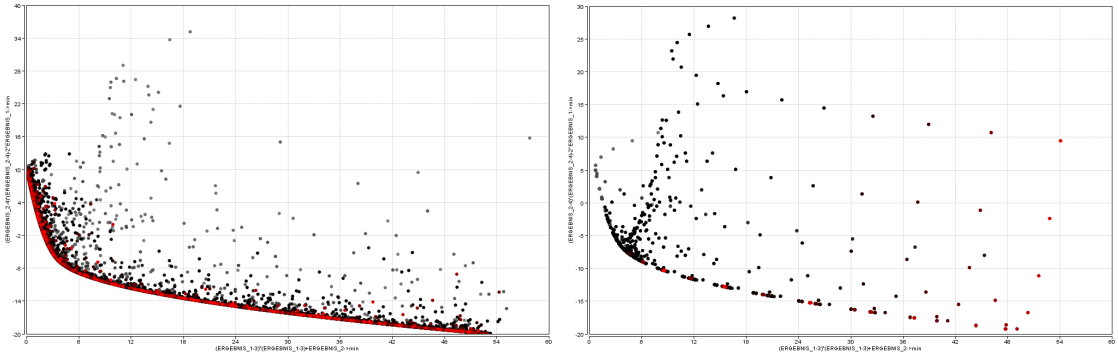
Bei dieser Testfunktion erreichen die Verfahren häufig das Minimum. Liegt der Startpunkt in der Nähe eines lokalen Minimums kann es vorkommen, dass die Verfahren dort konvergieren.

Zusammenfassend lässt sich sagen, dass keines der Verfahren in jeder Situation das globale Optimum findet. Gerade bei gradientenbasierten Verfahren ist es daher wichtig, die erhaltenen Lösungen kritisch zu betrachten und Kontrollrechnungen mit anderen Startpunkten anzustreben.

4.2 Multikriterielle Testfunktionen

Einfache konvexe Paretofront

$$
\begin{aligned}
f_1(x_1, x_2) &= (x_1 - 3)^2 + x_2 \\
f_2(x_1, x_2) &= (x_2 - 4)^2 - 2x_1 \\
x_1, x_2 &\in [0, 10]
\end{aligned}
$$



(a) SPEA (b) MGDA

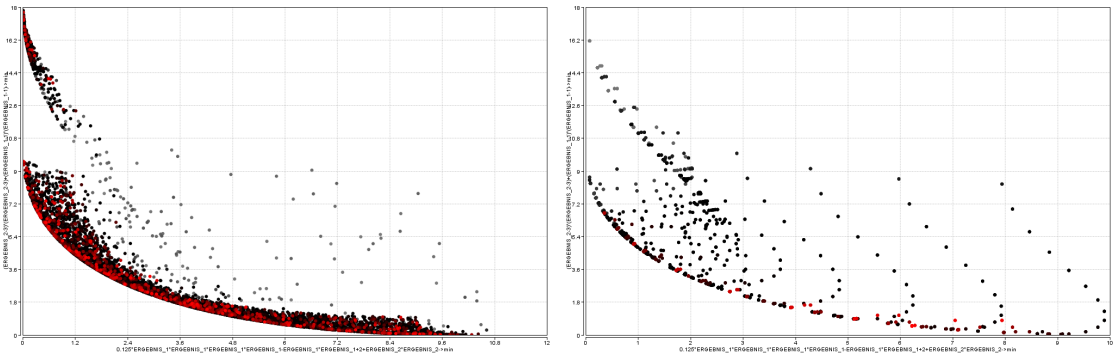
Abbildung 4.6: Konvexe Pareto-Front

Verfahren	SPEA2	IBEA	MGDA
erzeugte Varianten	6700	6700	128
Funktionsauswertungen	6700	6700	6719
paretooptimale Varianten	2297	3577	128
Anteil an Paretomenge der Vereinigung	0.7048	0.9502	1
mittlerer Abstand von der Pareto-Front	0.0435	0.0212	$3.0195 \cdot 10^{-7}$

Bei dieser Aufgabe zeigen sich die Stärken des MGDA. Alle 128 erzeugten Punkte sind in der Vereinigung der Pareto-Mengen enthalten. Das ist ein Indiz dafür, dass die Lösungen des MGDA die Lösungen von SPEA und IBEA lokal dominieren. Aufgrund der Einfachheit des Beispiels ist es über die analytische Lösung möglich, einen mittleren Abstand der Punkte zur Pareto-Front anzugeben. Bezüglich dieses Kriteriums ist der MGDA den evolutionären Algorithmen weit überlegen. In zusätzlichen Tests ergaben weitere Vergrößerungen der Punktmenge von SPEA und IBEA keine signifikanten Verbesserungen bezüglich des mittleren Abstands zur Pareto-Front.

lokale Pareto-Front

$$f_1(x_1,x_2) = \frac{x_1^4}{8} - x_1^2 + 2 + x_2^2$$
$$f_2(x_1,x_2) = (x_1-1)^2 + (x_2-3)^2$$
$$x_1 \in [-2,2]$$
$$x_2 \in [0,3]$$



(a) SPEA (b) MGDA

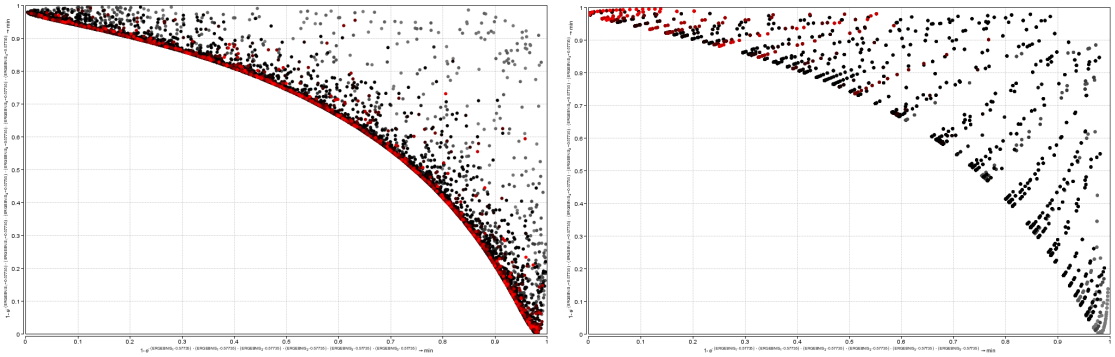
Abbildung 4.7: Funktion mit lokaler Pareto-Front

Verfahren	SPEA2	IBEA	MGDA
erzeugte Varianten	6700	6700	128
Funktionsauswertungen	6700	6700	5032
paretooptimale Varianten	808	934	103
Anteil an Paretomenge der Vereinigung	0.6955	0.7923	1

Dies ist die Aufgabe aus Beispiel 3.8. Wie zu erwarten konvergiert der MGDA von Startpunkten nahe der lokalen Pareto-Front gegen diese. Daher werden in diesem Beispiel auch nur 103 paretooptimale Varianten gefunden. Diese wiederum sind bezüglich ihrer Genauigkeit den Lösungen der anderen Verfahren überlegen. Im linken Bild der Abbildung 4.7 erkennt man, dass auch einige Lösungen der evolutionären Algorithmen zur lokalen Pareto-Front tendieren.

Fonseca Fleming Funktion

$$
\begin{aligned}
f_1(x_1, x_2, x_3) &= 1 - e^{-A} \\
f_2(x_1, x_2, x_3) &= 1 - e^{-B} \\
A &= \sum_{i=1}^{3} \left(x_i - \frac{1}{\sqrt{3}} \right)^2 \\
B &= \sum_{i=1}^{3} \left(x_i + \frac{1}{\sqrt{3}} \right)^2 \\
x_1, x_2, x_3 &\in [-4, 4]
\end{aligned}
$$



(a) SPEA (b) MGDA

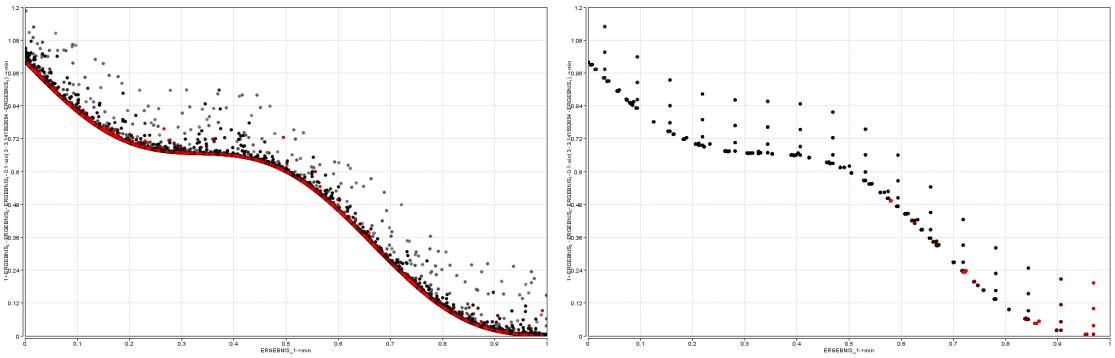
Abbildung 4.8: Fonseca Fleming Funktion

Verfahren	SPEA2	IBEA	MGDA
erzeugte Varianten	6700	6700	128
Funktionsauswertungen	6700	6700	8703
paretooptimale Varianten	1386	1732	128
Anteil an Paretomenge der Vereinigung	0.5765	0.9053	1

Bei dieser Testfunktion zeigt der MGDA keinerlei Probleme im Umgang mit der konkaven Pareto-Front. Erneut sind alle 128 erzeugten Punkte in der Vereinigung der Pareto-Mengen enthalten und somit lokal besser als die Lösungen von SPEA und IBEA. Die Punktedichte der evolutionären Verfahren ist mit dem MGDA jedoch nicht in einer angemessenen Berechnungszeit erreichbar.

Konvex-konkave Pareto-Front

$$
\begin{aligned}
f_1(x_1, x_2) &= x_1 \\
f_2(x_1, x_2) &= 1 + x_2^2 - x_1 - \sin(3\pi x_1) \\
x_1 &\in [0, 1] \\
x_2 &\in [-1, 1]
\end{aligned}
$$



(a) SPEA (b) MGDA

Abbildung 4.9: Konvex-konkave Pareto-Front

Verfahren	SPEA2	IBEA	MGDA
erzeugte Varianten	6700	6700	128
Funktionsauswertungen	6700	6700	6233
paretooptimale Varianten	2977	5169	127.98
Anteil an Paretomenge der Vereinigung	0.6957	0.9975	0.9998

Auch bei dieser Aufgabe liefert MGDA gute Ergebnisse. Bei einem der 50 Durchläufe wurde eine Lösung des MGDA von der Ergebnissen des IBEA dominiert. Der Wechsel von konkaven zu konvexen Bereichen der Pareto-front hat keinen erkennbaren Einfluss auf die Verteilung der Lösungspunkte des MGDA.

Insgesamt zeigen die multikriteriellen Tests, dass der MGDA den evolutionären Algorithmen im Bezug auf Genauigkeit weit überlegen ist. Andererseits ist es dem MGDA mit vergleichbarem Berechnungsaufwand nicht möglich, die Pareto-Front mit einer so großen Punktedichte zu approximieren, wie es die evolutionären Algorithmen tun. Diese Erkenntnisse legen eine Kombination der Verfahren nahe, um die guten Eigenschaften jeder Verfahrensklasse zu nutzen.

5 Zusammenfassung und Ausblick

Die in dieser Arbeit vorgestellten gradientenbasierten Optimierungsverfahren wurden in der IAV Otimierungssoftware implementiert und an konkrete Anwendungsgebiete angepasst. Sie werden vorwiegend zur Optimierung technischer Aufgabenstellungen eingesetzt, da hier die Genauigkeit der Lösungen hohe Priorität hat.

Der in Abschnitt 2 vorgestellte BFGS Algorithmus erzielt sehr gute Ergebnisse bei der Bearbeitung von Problemen mit Box-Constraints und wenigen Nebenbedingungen, wie sie z.B. in der Getriebeberechnung auftreten. Die vorgestellte Liniensuche ist bezüglich ihrer Schrittweite nach oben unbegrenzt. Vor allem in den ersten Iterationsschritten und über sehr flachen Gebieten der Zielfunktion erweist sich dies als nützlich.

MGDA zeigt gute Konvergenzeigenschaften in der Nähe der Pareto-Front. Da der Iterationsverlauf durch die ermittelten Gradienteninformationen bestimmt und nicht gesteuert werden kann, garantiert eine gleichmäßige Verteilung der Startpunkte leider nicht für eine gleichmäßige Verteilung der Lösungen auf der Pareto-Front. An dieser Stelle sollten noch weitere Verbesserungen des Algorithmus angestrebt werden.

Häufig ist die Auswertung von Zielfunktionen bei in der Praxis auftretenden Optimierungsproblemen sehr aufwändig und benötigt einen Großteil der Rechenzeit. Daher steht für die Optimierung oft nur eine begrenzte Anzahl an Funktionsauswertungen zur Verfügung. Um diese bestmöglich zu nutzen, besteht die Möglichkeit, den in Kapitel 3 vorgestellten Multiple Gradient Descent Algorithm neben der direkten Anwendung auch zur Nachiteration in hybriden Optimierungsverfahren einzusetzen. Es wäre denkbar, die Pareto-Front zuerst mit einem evolutionären Verfahren anzunähern. Anschließend kann der Anwender einen Bereich von Interesse auswählen, dessen Lösungen durch den MGDA nachiteriert werden. Voraussetzung dafür ist, das MGDA Ungleichungsnebenbedingungen verarbeiten kann. Dies wurde im Zuge dieser Arbeit noch nicht realisiert und bedarf weiterer Untersuchungen.

Auch eine Parallelisierung des Algorithmus ist problemlos möglich. Iterationen mit verschiedenen Startpunkten wie in den Beispielen aus Kapitel 4 sind voneinander unabhängig und können vollständig parallel ausgeführt werden. Eine derartige Parallelisierung bringt einen enormen Gewinn bezüglich der Berechnungszeit.

Insgesamt kann eingeschätzt werden, dass der MGDA interessante Möglichkeiten im Bereich der multikriteriellen Optimierung und großes Potential für neue, innovative Ansätze bietet.

Literaturverzeichnis

[Alt02] Alt, W., Nichtlineare Optimierung: Eine Einführung in Theorie, Verfahren und Anwendungen, Vieweg-Verlag, Wiesbaden 2002.

[Beb10] Bebendorf, M., Einführung in die numerische Mathematik, Vorlesungsskript, Universität Bonn 2010, unter: http://www.yumpu.com/de/document/view/17660030/vorlesungsskript-bebendorf-universitat-bonn (abgerufen am 15.07.2013).

[Des09] Desideri, J.A., Multiple-Gradient Descent Algorithm (MGDA), Research Report No 6953 , Sophia Antipolis 2009, unter: http://hal.archives-ouvertes.fr/docs/00/39/22/28/PDF/RR-6953.pdf .

[Des12] Desideri, J.A., MGDA II: A direct method for calculating a descent direction common to several criteria, Research Report No 7922 , Sophia Antipolis 2012, unter: http://hal.inria.fr/docs/00/68/57/62/PDF/RR-7922.pdf .

[Des12-2] Desideri, J.A., MGDA Variants for Multi-Objective Optimization, Research Report No 8068 , Sophia Antipolis 2012, unter: http://hal.inria.fr/docs/00/73/28/81/PDF/RR-8068.pdf .

[Des13] Desideri, J.A., http://www-sop.inria.fr/ members/ Jean-Antoine.Desideri/ Files/ oberwolfach2013.pdf (abgerufen am 22.07.2013).

[DS87] Dennis, J.E. und Schnabel, R.B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations,S.199 ff, SIAM 1987.

[Fro04] Frommer, A., Numerische Methoden der nichtlinearen Optimierung, Vorlesungsskript, Bergische Universität Wuppertal 2004, unter: http://www-ai.math.uni-wuppertal.de/ frommer/en/Course_Manuscripts/NichtLinOpt.pdf (abgerufen am 22.08.2013).

[GK99] Geiger, C. und Kanzow, C., Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben, S.1, Springer-Verlag, Berlin 1999.

[Har07] Harzheim, L., Strukturoptimierung: Grundlagen und Anwendungen, S.44, Europa Lehrmittel Verlag, Haan-Gruiten 2007.

[JS00] Jarre, F. und Stoer, J., Optimierung, Springer-Verlag, Berlin 2000.

[Kux11] Kux, S., Hybride Optimierungsstrategien für komplexe technische Aufgabenstellungen, Masterarbeit, Hochschule Mittweida 2011.

[Mei08] Meister, A., Numerik linearer Gleichungssysteme, S.51 f, Vieweg-Verlag, Wiesbaden 2008.

[Obe12] Oberle, H.J., Quasi-Newton-Verfahren, Vorlesungsskript, Universität Hamburg 2012, unter: http://www.math.uni-hamburg.de/home/oberle/skripte/optimierung/optim09.pdf (abgerufen am 10.07.2013).

[Sch00] Schittkowski, K., Mathematische Grundlagen von Optimierungsverfahren, Vorlesungsskript, Universität Bayreuth, unter: http://www.ai7.uni-bayreuth.de/skript.pdf (abgerufen am 02.08.2013).

[Stö07] Stöcker, M., Untersuchung von Optimierungsverfahren für Rechenzeitaufwändige technische Anwendungen in der Motorenentwicklung , Diplomarbeit, Techniche Universität Chemnitz 2007.

[Wei10] Weickert, J., Kapitel 57: Extrema von Funktionen mehrerer Variabler, S.2, Vorlesungsskript, Uni-Saarland 2010, unter: http://www.mia.uni-saarland.de/Teaching/MFI1011/kap57.pdf (abgerufen am 01.07.2013).

[ZDD11] Zerbinati, A. und Desideri, J.A. und Duvigneau, R., Comparison between MGDA and PAES for Multi-Objective Optimization, Research Report No 7667 , Sophia Antipolis 2011, unter: http://hal.inria.fr/docs/00/60/54/23/PDF/RR-7667.pdf (abgerufen am 24.07.2013).

[Zem08] Zemke, J.P., Numerische Verfahren, S. 24, Vorlesungsskript, TU Hamburg-Harburg 2008, unter: http://www.tuhh.de/~matjz/work/lectures/numver/folien/kapitel05.pdf (abgerufen am 10.07.2013).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 15. September 2013